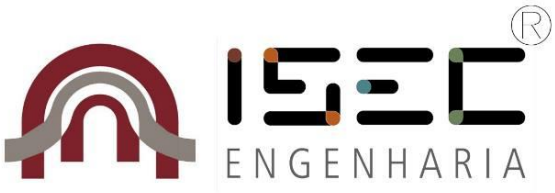

Mestrado em Informática e Sistemas

Implementação de uma infraestrutura de Cloud privada baseada em OpenStack

Dissertação apresentada para a obtenção do grau de Mestre em Informática e Sistemas
Especialização em Tecnologias da Informação e do Conhecimento

Autor
Tiago André Pais Rosado

Orientador
Prof. Doutor Jorge Bernardino
Professor do Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra

Supervisores
Paulo Rosado
MEO, Consultor IT
Délio Lourenço
MEO, Diretor IT

Coimbra, dezembro, 2016

"*Try not to become a man of success, but rather try to become a man of value.*"

Albert Einstein

AGRADECIMENTOS

Ao meu orientador por parte do ISEC, o Professor Doutor Jorge Bernardino, pessoa única com uma capacidade de trabalho e acompanhamento fora de série, sempre presente e acima de tudo, paciente.

Ao terminar este projeto não posso deixar de agradecer aos meus orientadores na Portugal Telecom, Paulo Rosado e Délio Lourenço por todo o apoio, ensinamentos e sugestões. Pessoas ímpares e de excelência, sem as quais não teria sido possível a realização deste projeto, bem como à empresa pelos meios disponibilizados para a realização deste estudo.

Ao Instituto Superior de Engenharia de Coimbra (*ISEC*), mais concretamente ao Departamento de Engenharia Informática e de Sistemas (*DEIS*) e respetivos docentes, por, ao longo do meu percurso académico me terem transmitido o conhecimento e dado as ferramentas para prosseguir na carreira profissional.

À Sandra, mulher da minha vida!

De uma forma particular agradeço aos pilares da minha vida, Deus, Família e Amigos, concretamente ao meu pai, mãe e irmã por todo o apoio incondicional.

Aos suspeitos do costume! Sim, aqueles, esses mesmos, os tais que continuam lá contigo ao longo do caminho mesmo estando desaparecido, isolado, incomunicável, os que não hesitam quando ouvem um grito de socorro, os que não são de sangue mas são família, amigos do coração.

A todos, os meus mais sinceros agradecimentos.

RESUMO

A constante evolução da Internet e o seu uso cada vez mais generalizado serviram de alavanca para a proliferação de novos serviços e arquiteturas que visam dar resposta à procura do setor privado mas principalmente à necessidade de otimização e exigência de evolução das empresas. Soluções de computação em nuvem, do inglês *Cloud Computing*, tem vindo a juntar-se ao vocabulário e ao dia-a-dia da área das Tecnologias de Informação e Comunicação e que geralmente se encontra associado a um modelo aplicacional de elevada flexibilidade onde um diversificado leque de recursos e serviços são entregues aos utilizadores finais por meio de uma ligação à Internet.

Partindo da premissa que o progresso e o aperfeiçoamento devem começar pela empresa, este trabalho tem como objetivo a implementação de uma infraestrutura privada de *Cloud Computing* utilizando *software open source* a fim de responder aos desafios colocados pelos órgão de gestão às áreas de *TIC* relativamente às questões de controlo de custos, em termos de aquisição de equipamentos e licenciamento, disponibilizando de forma mais ágil uma infraestrutura altamente personalizável mantendo os elevados padrões de qualidade, segurança e robustez.

A implementação de uma solução de *Cloud Computing* privada com recurso a *software open source*, o *OpenStack*, evita que a empresa recorra a soluções proprietárias, por vezes dispendiosas, e permitirá fornecer aos departamentos de pesquisa e desenvolvimento uma plataforma com os recursos pretendidos, por forma a torná-los mais autónomos e independentes na configuração e gestão de servidores sem que coloquem em risco a infraestrutura de *TIC* da empresa.

Na presente dissertação é possível ficar a conhecer a plataforma do *OpenStack*, os seus elementos constituintes e a forma como estes se interligam de modo a apresentar uma solução de *Cloud*, tendo esta experiência permitindo ainda demonstrar que é possível implementar uma infraestrutura de *Cloud Computing* fiável com recurso a *software open source*, equipamentos de gama doméstica e, ainda assim, obter bons níveis de desempenho.

A informação é muita e por vezes dispersa, o que, aliado ao interesse da empresa em explorar o tema da *Cloud* serviu de incentivo ao estudo da plataforma do *OpenStack* e das várias implementações possíveis. De modo a responder ao objetivo de como criar uma *IaaS* economicamente viável, tanto na utilização de *software open source* como no recurso ao *hardware*, foi feita uma implementação prática da solução onde se recorreu a equipamentos utilizados diariamente na empresa. O estudo apresentado na empresa ficou concluído com a avaliação de performance da solução.

PALAVRAS-CHAVE:

Cloud Computing; *Private Cloud Computing*; *OpenStack*; *Open source*; Infraestrutura de baixo custo; *IaaS* - Infraestrutura como um serviço;

ABSTRACT

The constant evolution of the Internet and its increasingly widespread use served as a lever for the proliferation of new services and architectures that aim to meet the demand of the private sector but mainly to the need for optimization and demand for development of enterprises.

Cloud Computing solutions has join the vocabulary and the day-to-day of the Information Technology area and usually is associated with an application model of high flexibility where a diverse range of resources and IT services are delivered to end users through an Internet connection.

Assuming that progress and improvement come from within, this work aims to implement a private cloud computing infrastructure using open source software in order to meet the challenges posed by the management body to IT departments regarding the cost control issues, in terms of acquisition of equipment and licensing, providing a more agile and highly customizable infrastructure while maintaining high standards of quality, safety and robustness.

The implementation of a private cloud computing solution using open source software prevents the firm from using proprietary solutions, sometimes costly, and will provide to the research and development departments a platform with the required resources in order to make them more autonomous and independent from server management and configuration without endangering the company's IT infrastructure.

This present work introduced us the OpenStack platform, its constituent elements and how they interrelate to present a Cloud solution, having this experience allowing to demonstrate that it is possible to implement a reliable Cloud Computing infrastructure using open source software, household range hardware and still obtain good performance levels.

The information is a lot and sometimes scattered, which, coupled with the company's interest in Cloud Computing, served as an incentive to study the OpenStack platform and the various possible implementations. In order to respond to the goal of how to create an economically viable IaaS, both in the use of open source software and in the use of hardware, a practical implementation of the solution was made using daily equipment used in the company. The study presented at the company was concluded with the performance evaluation of the solution.

KEYWORDS:

Cloud Computing; Private Cloud Computing; OpenStack; Open Source; Low Cost Infrastructure; IaaS - Infrastructure as a Service;

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

SIMBOLOGIA E ABREVIATURAS

AMI	Amazon Machine Image
AMQP	Advanced Message Queue Protocol
API	Application Programming Interface
AKI	Amazon Kernel Image
ARI	Amazon Ramdisk Image
AWS	Amazon Web Service
BSD	Berkeley Software Distribution
CaaS	Comunications as a Service (Comunicações como um serviço)
CAPEX	Capital Expenditures
CC	Eucalyptus Cluster Controller
CLC	Eucalyptus Cloud Controller
CPU	Central Processing Unit
dBaaS	Database as Service (Base de Dados como um Serviço)
DEIS	Departamento de Engenharia Informática e de Sistemas
DFSG	Debian Free Software Guidelines
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
EBS	Amazon Elastic Block Store
EC2	Elastic Compute Cloud (Amazon Web Service)
Eucalyptus	Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems
FSF	Free Software Foundation
GNU	GNU is Not Unix
GPL	GNU General Public Licence
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service (Infra-estrutura como um Serviço)
ISEC	Instituto Superior de Engenharia de Coimbra
IDC	International Data Corporation
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LTS	Long Term Support
MaaS	Monitoring as a Service (Monitorização como um Serviço)
MIT	Massachusetts Institute of Technology
NC	Eucalyptus Node Controller
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
OPEX	Operating Expenses

OSI	Open Source Initiative
PaaS	Platform as a Service (Plataforma como um serviço)
PME	Pequenas e Médias Empresas
QCOW	QEMU Copy On Write
RAID	Originalmente Redundant Array of Inexpensive Disks, Atualmente Commonly Array of Independent Disks
RAM	Random-access Memory
ReST	Representational State Transfer
RHEL	Red Hat Enterprise Linux
S3	Amazon Simple Storage Service
SaaS	Software as a Service (Software como um serviço)
SC	Eucalyptus Storage Controller
SFF	Small Form Factor
SNS	Amazon Simple Notification Service
SQS	Amazon Simple Queue Service
TIC	Tecnologias de Informação e Comunicação
VDI	VirtualBox Disk Image (formato de uma imagem)
VDI	Virtual Desktop Infrastructure (Equipamento físico, terminal *dummy*)
VGrADS	Virtual Grid Application Development Software
VHD	Virtual Hard Disk
VLAN	Virtual Local Area Network
VMDK	Virtual Machine Disk
VPN	Virtual Private Network
XaaS	"Anything as a Service" ou "Everything as a Service" (Tudo como um Serviço)

1. INTRODUÇÃO

Tudo que envolve tecnologia está sempre em mudança e evolução constantes, onde a área das Tecnologias de Informação e Comunicação (TIC) não é exceção. Cada dia que passa, fruto de uma comunidade maior e mais exigente, são lançadas novas aplicações, novos meios de encarar problemas antigos, novas soluções para questões do quotidiano ou para a reinvenção deste.

O *software open source* tem acompanhado a evolução do meio em que se enquadra e conta, para além de uma comunidade motivada, com a presença forte e manifestamente interessada de parceiros tecnológicos de diversas áreas, onde se incluem grandes empresas de desenvolvimento de software proprietário.

Como resultado e consequência da enorme facilidade de acesso à Internet, surge o conceito de computação em nuvem, do inglês *Cloud Computing* (termo adotado ao longo desta dissertação, fruto da sua maior popularidade). Este termo tem vindo a juntar-se ao vocabulário e ao dia-a-dia da área das TIC – onde tem vindo a ganhar cada vez mais atenção nos últimos anos, tendo-se tornado um tema de substancial interesse empresarial e académico pois é muitas vezes aplicado a um modelo aplicacional de elevada flexibilidade onde um diversificado leque de recursos e serviços de TIC são entregues aos utilizadores finais por meio de uma ligação à Internet, também conhecido por soluções que disponibilizam infraestruturas como um serviço, ou pela sua abreviatura do inglês, *IaaS* – e o seu conceito vem satisfazer uma necessidade crescente de serviços de TIC feitos à medida, sem que haja necessariamente um investimento direto em nova infraestrutura, formação de novos técnicos ou licenciamento de *software*, reduzindo assim o tempo de resposta face ao mercado impondo um ritmo de inovação mais elevado.

A abertura das empresas a novas soluções é cada vez maior, no entanto, deparam-se com constrangimentos de ordem financeira para investimentos de fundo ou pela falta de informação concreta, objetiva e de qualidade. Foram estas as premissas de base de trabalho para esta dissertação, ser capaz de concentrar a informação existente sobre o tema, apresentar uma solução robusta e flexível e deixar conhecimento para que fosse possível progredir.

Aliado à documentação sobre a plataforma, foi concretizado o estudo com uma prova de conceito, validando dessa forma a solução do *OpenStack*. Sendo a infraestrutura do *OpenStack* uma *IaaS*, em que a base de ação é a disponibilização de máquinas virtuais, foi complementado este trabalho com a apresentação de uma análise sobre os parâmetros que podem influenciar a sua performance.

1.1. Motivações e Objetivos

As empresas de TIC estão cada vez mais focadas na vanguarda da tecnologia, desenvolvendo novas plataformas e soluções de negócio, tendo assim uma necessidade constante de recursos para apoio à sua atividade. Esses recursos podem ser armazenamento, servidores de qualidade para teste de uma solução antes de esta ser lançada para o mercado,

servidores para realizar provas de conceito ou plataformas para teste de módulos de *software*, o que implicará configurações diferentes, específicas para cada projeto, equipa de desenvolvimento ou departamento. Estar dependente do departamento de TIC para o aprovisionamento à medida de uma variedade enorme e altamente mutável de configurações, tanto de *hardware* como de *software*, consome tempo que qualquer uma das equipas não dispõe. Enquanto por um lado as diversas áreas de TIC se debatem para disponibilizar os recursos e serviços necessários para fazer face às exigências do mercado, por outro, os órgãos de gestão estão focados no controlo de custos e investimentos.

Partindo da premissa que o progresso e o aperfeiçoamento devem começar pela empresa, este trabalho tem como objetivo a implementação de uma infraestrutura privada de *Cloud Computing* utilizando *software open source* a fim de responder aos desafios colocados pelos órgão de gestão às áreas de TIC relativamente às questões de controlo de custos, em termos de aquisição de equipamentos e licenciamento, disponibilizando de forma mais ágil uma infraestrutura altamente personalizável mantendo os elevados padrões de qualidade, segurança e robustez. A implementação de uma solução de *Cloud Computing* privada com recurso a *software open source* evita que a empresa recorra a soluções proprietárias, por vezes dispendiosas, e permitirá fornecer aos departamentos de pesquisa e desenvolvimento os meios necessários por forma a torná-los mais autónomos e independentes na configuração e gestão de servidores sem que coloquem em risco a infraestrutura de TIC da empresa.

Por outro lado, esta experiência vai permitir demonstrar que é possível implementar uma infraestrutura de *Cloud Computing* fiável com recurso a *software open source*, equipamentos de gama comercial e ainda assim, obter bons níveis de desempenho, e mais uma vez, ir ao encontro da redução de custos.

Como síntese podemos identificar os seguintes pontos-chave que foram propostos e ao que esta dissertação e trabalho realizado pretendem responder:

- Identificar soluções de *Cloud open source* fiáveis, válidas e com alguma maturidade.
- Estudar e apresentar uma solução de modo a que o departamento que fique com a continuação do trabalho consiga ter uma base sólida para manter ou evoluir.
- Elaborar uma prova de conceito com a solução de Cloud estudada de modo a perceber a validade para colocar um projeto em produção.
- Realizar um estudo de performance da infraestrutura no que concerne à disponibilização de servidores virtuais que contemplem diferentes cargas de recursos associados.

1.2. Principais contributos deste trabalho

Tendo em vista a contribuição para o avanço da ciência e do conhecimento, foi identificado o problema, que se prende com o fato de otimização de meios e recursos para uma melhor eficiência das equipa de TIC das diversas áreas funcionais de uma empresa e a credibilização de soluções *open source*, assim os principais contributos deste trabalho são:

- Mostrar que a plataforma *OpenStack* é uma alternativa altamente viável face às soluções comerciais existentes.
- Mostrar que o *open source* é uma opção viável no contexto empresarial.
- Contribuir para o aumento da competitividade e autonomia das diversas áreas de TIC de uma empresa.
- Uma solução de *Cloud* privada *open source* contribui para o controlo de custos de hardware e licenciamento.

De uma forma geral, foram estas as principais contribuições dadas com o desenvolvimento deste trabalho.

De salientar também que, como fruto deste trabalho, foram elaborados e publicados os seguintes artigos científicos, nomeadamente:

- "*A low cost private cloud infrastructure using OpenStack*" na conferência "IEEE BigData 2014 Coimbra Satellite Session", Tiago Rosado e Jorge Bernardino.
- "*An overview of OpenStack architecture*", Tiago Rosado, Jorge Bernardino. IDEAS 2014, 18th International Database Engineering & Applications Symposium, IDEAS 2014, Porto, Portugal, July 7-9, 2014. ACM 2014, ISBN 978-1-4503-2627-8, pp. 366-367.
- "*Implementation of a Low Cost IaaS using OpenStack*", Tiago Rosado, Jorge Bernardino. ICSOFT-EA 2016, Proceedings of the 11th International Joint Conference on Software Technologies (ICSOFT 2016) - Volume 1: ICSOFT-EA, Lisbon, Portugal, July 24 - 26, 2016. SciTePress 2016, ISBN 978-989-758-194-6, pp. 298-303.

De referir que o artigo publicado na conferência IDEAS'14 conta atualmente com 21 citações no Google Académico:

https://scholar.google.pt/citations?user=EpJW32AAAAAJ&hl=pt-PT

1.3. Estrutura do Relatório

Esta dissertação de mestrado encontra-se estruturada em 8 capítulos. No presente capítulo, introdutório, deu-se a conhecer o problema, tendo sido apresentadas as principais contribuições deste trabalho e a estrutura do relatório.

O capítulo 2 descreve o *software open source*, como este evoluiu e se apresenta atualmente, fazendo referências a algumas licenças, qual o impacto da sua utilização nas empresas, sendo dado exemplos de software de sucesso.

No capítulo 3 é introduzido o tema *Cloud Computing*, apresentando o que se considera ser a sua definição e principais características. São também analisados os principais modelos de serviço e de implementação.

O capítulo 4 apresenta uma análise de cinco plataformas *open source* de *Cloud Computing* focando-se nas suas principais características para que possa ser feita uma comparação entre elas.

O capítulo 5 está focado na apresentação e detalhe da plataforma sobre a qual incide este estudo, o *OpenStack*. É apresentado o seu modelo conceptual e lógico, o modo como se agrupam logicamente os vários módulos constituintes da plataforma, seguido da apresentação da arquitetura de serviços que nos mostra como se relacionam em prol do funcionamento global da plataforma, terminando com a apresentação do modelo de objetos e a sua hierarquia.

O capítulo 6 é dedicado à implementação desta plataforma, focando os módulos principais, abordados no capítulo anterior, de modo a disponibilizar uma infraestrutura de *Cloud Computing*. São apresentadas as configurações utilizadas em termos de hardware e software.

No capítulo 7 é apresentada a avaliação experimental da plataforma implementada.

Por fim, no capítulo 8 desta dissertação são apresentadas as principais conclusões do trabalho realizado sendo ainda proposto algum trabalho futuro.

Como anexos foram produzidos os seguintes documentos:

Anexo A: "*A low cost private cloud infrastructure using OpenStack*" publicado na conferência IEEE BigData 2014 Coimbra Satellite Session.

Anexo B: "*An overview of OpenStack architecture*" publicado na conferência 18th International Database Engineering & Applications Symposium, IDEAS '14P.

Anexo C: "Implementation of a Low Cost IaaS using OpenStack" publicado na conferência 11th International Joint Conference on Software Technologies (ICSOFT 2016).

Anexo D: Criação de imagens *Windows* com instalação de drivers *virtIO*.

Anexo E: Instalação da ferramenta RALLY.

Anexo F: Ambiente gráfico do OpenStack.

Anexo G: Tempos recolhidos para análise de performance.

2. SOFTWARE OPEN SOURCE

É comum existir em alguns trabalhos a ideia de que *software open source* e *software* livre como sendo o mesmo e, embora não seja necessariamente errado, existem várias diferenças. *Software* livre é um conceito de extrema importância na realidade das tecnologias de informação, e para ser considerado como tal tem de obedecer a determinados parâmetros e características de liberdade, podendo dizer-se que este é um movimento social, que defende uma causa. A ideia ganhou vida pelas mãos de *Richard Stallman*, criador do *GNU* (acrónimo recursivo de "*GNU is Not Unix*") e fundador da *FSF* (*Free Software Foundation*), uma entidade sem fins lucrativos criada para servir de base ao movimento do *software* livre (ver logotipo na Figura 2.1). A liberdade e os seus princípios, segundo os quais o movimento se rege e que já tinham sido o alicerce para a criação do *GNU*, assentam em quatro fundamentos base (*FSF, 2014*):

- Liberdade de executar o programa;
- Liberdade de estudar como o programa funciona e adaptá-lo às necessidades;
- Liberdade de redistribuir cópias;
- Liberdade de aperfeiçoar o programa e fazer melhorias publicamente disponíveis, para benefício da comunidade.

De notar que *software* livre não quer dizer necessariamente *software* gratuito, um *software* pode ser vendido desde que não entre em rota de colisão com as bases da *FSF*; do mesmo modo que, um *software* gratuito em que não haja a necessidade de pagar uma licença, é significado de *software* livre ou *open source*, pois um *software* proprietário pode ser disponibilizado de forma gratuita e no entanto não ser possível alterar, melhorar ou utilizar-se para todos os fins pretendidos.

Figura 2.1 - Logotipo da *Free Software Foundation* (*FSF, 2014*)

O *Open source* por seu lado é um movimento que surgiu em 1998, encabeçado por *Eric Raymond* e *Bruce Perens*, juntamente com um grupo enorme e variado de pessoas que não se reviam nos ideais filosóficos, éticos e efeitos sociais relacionados com o *software* livre, considerando-os como um obstáculo para uma rápida aceitação por parte dos investidores e empresários, o que resultou assim na criação da *OSI* (*Open Source Initiative*). A *OSI* (ver logotipo na Figura 2.2), organização de utilidade pública sem fins lucrativos, criada com o intuito de educar e defender a utilização de *software* de código aberto (tradução do inglês, *open source*) e responsável pela revisão e aprovação de licenças *open source*, não ignora as liberdades da FSF, por outro lado, tenta ser mais flexível e para tal, definiu dez requisitos para que o *software* possa ser considerado *open source*:

- Distribuição livre;
- Acesso ao código fonte;
- Permissão para criação de trabalhos derivados;
- Integridade do autor do código fonte;
- Não discriminação contra pessoas ou grupos;
- Não discriminação contra áreas de atuação;
- Distribuição da licença;
- Licença não específica a um produto;
- Licença não restritiva a outros programas;
- Licença neutra em relação à tecnologia.

Os requisitos identificados defendem que este tipo de *software* pode ser copiado, executado, distribuído, modificado e aperfeiçoado por todos os utilizadores ambicionando dessa forma uma melhor qualidade, confiabilidade, mais flexibilidade e menor custo (*Open Source, 2014*).

Figura 2.2 - Logotipo da *Open Source Initiative* (*Opens Source, 2014*)

Ao analisar as características de ambos os tipos de *software* chega-se à conclusão que, em muitos casos, um software livre também pode ser considerado de código aberto (tradução do inglês, *open source*) e vice-versa, onde a diferença assenta essencialmente no facto da OSI ter uma maior recetividade relativamente às iniciativas de mercado por parte de empresas de *software* proprietário que pretendam desenvolver soluções *open source*, desde que estas respeitem os seus critérios; já na ótica do *software* livre, tais empresas poderiam vir a sofrer alguma resistência pelo facto dos princípios da sua atividade e da oferta que tenham para o mercado poderem entrar em conflito com os ideais morais da *FSF*.

Embora não concordem em alguns pontos, ambas as entidades são necessárias e complementam-se, enquanto a *FSF* por vezes com um discurso mais politizado e considerado de certa forma radical, atua sob a vertente mais social, a *OSI* sob os contextos técnicos e de mercado.

Em termos do desenvolvimento de *software*, o método utilizado e defendido por cada uma das entidades, *FSF* e *OSI*, distanciam-se na sua essência. *Raymond*, que fazia parte de uma nova geração de programadores e defensor do *open source*, critica *Stallman* na sua publicação "A

Catedral e o Bazar" (*Raymond, 1997*) onde compara o desenvolvimento do *GNU*, bem como da maioria dos projetos da *FSF*, a uma Catedral, monumentos sólidos desenvolvidos a partir de um grande planeamento, onde um grupo de arquitetos exerce direção e controlo sobre os trabalhadores, sendo este também o modelo do desenvolvimento de software proprietário. Já o modelo adotado por *Linus Torvalds* é comparado a um bazar onde não existe uma hierarquia entre os participantes, criando-se uma dinâmica bastante descentralizada onde as alterações ao *software* são disponibilizadas rapidamente para que a comunidade analise, avalie e proponha novas soluções e correções. Conforme enunciado por *Rymond*, "Dados olhos suficientes, todos os erros são óbvios"[1] (Raymond, 1997).

A categorização do *software* disponibilizado no mercado, seja ele de um modo geral, livre, *open source* ou proprietário, só é possível devido às licenças que os acompanham e que determinam o seu grau de liberdade, tendo em conta as quatro liberdades básicas, a cópia o uso a modificação e a redistribuição. Na secção que se segue são explicadas algumas das licenças de *software* mais populares.

2.1. Licenças de Software

Ao ser disponibilizado um *software*, os seus autores definem o grau de liberdade com que modificações e redistribuições podem ser efetuadas e é uma licença de software que define essas liberdades de uma forma mais ou menos restritiva. A Figura 2.3 mostra as interseções entre as várias categorias de licenças de *software (Categories of free and nonfree software, 2014*).

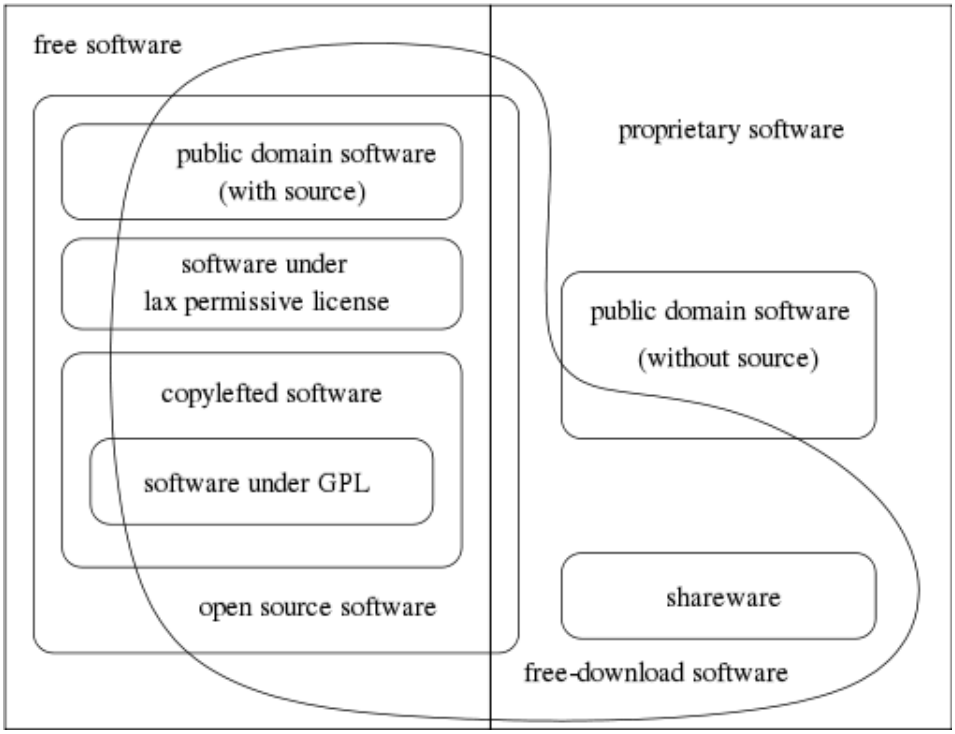

Figura 2.3 - Categorias de licenças de *software*

[1] Http://www.unterstein.net/su/docs/CathBaz.pdf

A maioria das licenças que acompanham o *software* livre permitem a modificação e redistribuição dos programas, práticas estas que são geralmente proibidas pela legislação internacional do *copyright*, que tenta justamente impedir que se efetuem tais ações sem a autorização do(s) autor(es). De modo a não infringir a legislação foi criada uma versão do *copyright*, o *copyleft*, onde é determinado que qualquer trabalho derivado precisa ser distribuído sob os mesmos termos da licença original. Desta forma são definidas explicitamente as condições sob as quais as cópias, alterações e distribuições podem ser feitas por forma a garantir as liberdades de modificar e distribuir o *software* assim licenciado (*What is Copyleft, 2014*). Podemos ver na Figura 2.4 os logotipos do *copyright* e do *copyleft* respetivamente.

Figura 2.4 - Da esquerda para a direita, o logotipo do *Copyright* seguido do *Copyleft*

Qualquer licença traduz em si as liberdades que fazem com que o *software* pertença a um dos modelos de desenvolvimento falado anteriormente. A licença mais popular utilizada em *software open source* é a *GPL* (*GNU General Public Licence*), publicada pela *FSF*, no entanto, não existem impedimentos para que sejam criadas novas licenças, desde que sejam garantidas as quatro principais liberdades. De seguida são apresentadas algumas das licenças mais populares.

GPL – Significa *GNU General Public Licence* e é a licença que acompanha os desenvolvimentos feitos pelo projeto *GNU*, onde se inclui o sistema operativo *Linux*. Esta licença dá toda a liberdade de cópia, alteração e distribuição do *software* por ela protegido bem como a segurança no sentido em que não podem ser adicionadas restrições às descritas na licença. Assim, como exemplo, impede que *software* licenciado com GPL possa ser utilizado em *software* proprietário (*GNU General Public Licence, 2014*).

BSD – Foi a primeira licença de *software* livre escrita, criada originalmente pela Universidade da Califórnia em *Berkeley* para o seu sistema operativo derivado do *Unix*, o *Berkeley* Softwa*re Distribution*. Estas licenças são consideradas permissivas, também chamadas de licenças académicas por alguns autores, pois permitem quase tudo desde que *a priori* o autor original seja citado. Não existe qualquer obrigatoriedade quanto ao código fonte e ao contrário da licença *GPL*, *software* com esta licença pode ser incorporado em *software* proprietário (*BSD License Definition, 2014*).

MIT/X11 – Licença criada pelo *Massachusetts Institute of Technology* (*MIT*), também conhecida como Licença *X11* ou *X* derivado de ter sido criada para o *X Window System*, criado pelo *MIT* em 1987. Esta é também uma licença permissiva, do género da *BSD*, no entanto ainda menos restritiva, mais simples e a menor de todas, restringindo apenas que o *software* deverá ser acompanhado da licença.

Apache – A licença Apache também é permissiva e é utilizada por um projeto bastante conhecido do software livre, o servidor *Web Apache*, assim como pela maioria dos outros

projetos pertencentes à fundação *Apache*. Para além de a licença ter de ser mantida caso alguém modifique o produto bem como dados os devidos crédito, são ainda colocadas algumas condições para a redistribuição do trabalho e seus derivados, tais como: os direitos da licença são perpétuos, são garantidos mundialmente, são gratuitos e sem qualquer tipo de *royalties*, não são exclusivos e os direitos não podem ser retirados, sendo irrevogáveis (*The Apache Software Foundation, 2014*). Como curiosidade, esta licença é usualmente utilizada pela *Google* nos seus produtos *open source*.

Debian – Esta licença, também conhecida e chamada de *Debian Free Software Guidelines* (*DFSG*), é parte do contrato social celebrado com a comunidade de *software open source* e que cumpre com os critérios da licença *GPL*, onde se salienta a diferença no sentido em que a licença não deve "contaminar" outro software, ou seja, não deve insistir para que todos os programas distribuídos tenham de ser livres (*Debian Social Contract, 2014*).

Para além deste tipo de licenças, destacam-se algumas que, embora o nome o sugira não são *software* livre, que é o caso das licenças *Freeware* (*Freeware, 2014*) – que significa software gratuito mas não livre como o free software – e as *Sharewhare* (*Shareware, 2014*) que permite a sua distribuição mediante o pagamento da licença no entanto não permite a distribuição do código, inviabilizando desta forma quaisquer alterações.

No subcapítulo que se segue, irá ser abordado o tema da utilização do *open source* nas empresas enunciando algumas das suas vantagens e desvantagens.

2.2. Utilização de Open Source nas Empresas

Quando se pretende implementar um novo *software* no seio de uma empresa há vários fatores relevantes a ter em consideração aquando da tomada dessa decisão; fatores esses que passam pela avaliação do impacto na produtividade da empresa e todo um conjunto de custos, desde o licenciamento, ao de aquisição, manutenção e suporte ou de formação dos utilizadores que deverão ser equacionados para uma decisão informada, no entanto, o desconhecimento de alternativas ou a pressão da administração podem levar as empresas a tomadas de decisão que se podem revelar contraproducentes.

O constante evoluir do meio tecnológico e claramente nos últimos anos do *software open source*, em termos de oferta e mais ainda em termos de qualidade, faz dele uma opção viável, credível e de enorme potencial que deveria ser considerada pelas empresas, por qualquer decisor ou técnico envolvido de alguma forma com as tecnologias de informação (*Baptista et al., 2004*).

Se por um lado os baixos custos são um fator importante na adoção do *open source*, a fraca documentação, a falta de apoio e o profundo desconhecimento já foram outrora a principal recusa, o que mesmo assim não faz com que deixem de existir cada vez mais e maiores casos de sucesso em ambiente empresarial como é o exemplo na Europa da *Airbus* ou o *Deutsche Bank*, entre outros, e mais concretamente em Portugal, o próprio Estado Português, a Brisa, Via Verde ou a AIRC, esta última não só pela adoção como pelo desenvolvimento de uma plataforma de *Business Intelligence* (Casos de Sucesso, 2014).

Não se pode ser radical ao ponto de considerar o *software* proprietário um mal necessário ou de o querer erradicar devido há dependências dos fornecedores relativamente ao suporte e

atualizações ou atendendo aos custos das licenças porque nem tudo no mundo do *open source* são maravilhas, existem, tal como no seio do *software* proprietário, as suas lacunas e este deve ser estudado e avaliado para cada situação em concreto, no entanto estes dois mundos podem e devem de fato coabitar no melhor interesse das pessoas, empresas e instituições envolvidas.

Existem de fato vantagens mais do que comprovadas do mesmo modo que se conseguem identificar desvantagens na sua utilização e que na altura de optar por uma solução deverão ser consideradas (*Steyn, 2012*; *Sinfic, 2008*). Como principais vantagens para o *open source* podem-se referir:

- Baixo custo ou valor nulo para as licenças.
- Atualizações constantes por parte da comunidade.
- Comunidades fortes e empenhadas na partilha de informação, esclarecimento de dúvidas e ajuda na resolução de problemas.
- Forte adaptabilidade das soluções devido à sua modularidade.
- Fácil integração em sistemas legados.
- Pode reduzir a dependência do fornecedor e/ou tecnológica.

Como principais desvantagens encontramos:

- Interfaces pouco amigáveis.
- Inexistência de vínculos contratuais podendo induzir alguma insegurança à empresa que vai adotar a solução.
- Instalação e configuração necessitam de técnicos especializados.
- Maior dificuldade em implementar módulos à medida.
- Inexistência de uma versão estável devido à constante evolução de módulos pela comunidade.
- Falta de suporte oficial.

A variedade nem sempre é algo benéfica no momento de fazer uma escolha e quando se trata de *software open source* existe de fato um leque vasto de opções que se podem tomar. Podemos chegar a essa conclusão com uma simples pesquisa em repositórios de *software* online em que o retorno de uma pesquisa pode ascender facilmente a 100 resultados. Por forma a minimizar a lista de resultados há alguns indicadores que poderão ajudar no momento da escolha, tais como, indicadores de atividade, idade do projeto, data da última atualização, número de mensagens nos fóruns do projeto, existência ou não de uma página oficial do projeto atualizada ou quantidade de resultados nos motores de pesquisa.

Há muitos exemplos notáveis de *software open source* que já conseguiu penetrar e mostrar a sua robustez e competitividade no mercado, no entanto todos os dias surgem novas soluções, novos projetos. A Tabela 2.1 é representativa disso mesmo, mostrando exemplos em diversos domínios.

Esta não é uma representação exaustiva pelos motivos anteriormente referidos (Lista de softwares *open source* para *Windows*).

Tabela 2.1 - Exemplos de *software open source* em diversos domínios

DOMÍNIO	SOFTWARE
Modelação de dados	Argo UML, Open Ameos, Jude
Antivirus e Manutenção	ClamWin, MemTest
Partilha de ficheiros	Shareaza, Freenet, Azureus, BitTorrent, eMule, LimeWire, Ares, Deluge
Desenvolvimento	Euphoria, Code::Blocks, Dev-C++, Dev Pascal, Eclipse, Free Pascal, Icon, Lazarus, Lua, MinGW, Netbeans, Perl, Python, Quincy, Ruby, Ruby on Rails, Sharp Develop, Unicon, CA-Realia, qt
Edição de Imagem	Paint.NET, Inkscape, Gimp
Modelação 3D	Blender, Wings 3d
Edição de Som e Vídeo	Aimp2, Audacity, Virtualdubmod, MPlayer, Media player classic, VLC, Songbird, Miro, Lmms
Emuladores	Stella,O2EM, MEKA, Gens, ZSNES, Finalburn Alpha, Project64, Visual Boy Advance, Nintendo DS
Gestão de Downloads	Free Download Manager, TrueDownloader, Orbit, Millweed, Tor
Mensagens Instantâneas	Pidgin, aMSN, Miranda IM, Mercury IM
Jogos	FreeCol, LGames, SpaceShooter, Frets on Fire, Battle for Wesnoth, UFO Alien Invasion, Quake 1, 2 e 3
Browser Email e Calendário	Google Chrome, Mozilla Firefox, Mozilla Thunderbird, Mozilla Sunbird, K-Meleon, SeaMonkey
Utilitários	1-4a Rename, 7-zip, Filzip, Vorbis, wxMusik, Filezilla
Office	BROffice.org, LibreOffice, Crimson Editor, DeskLight
Software Científico	SciLab, Octave, Maxima, FreeMat, JMathLib
Gestão Documental	GED Maarch

Conforme já referido anteriormente, o *software open source* encontra-se em constante mudança, e a informação acima identificada pode ser encontrada de forma atualizada em repositórios online, de onde se destacam o *http://ww1.sourcefourge.net/*, o *http://opensource.org/* ou o *http://www.w3.org/Status*.

Neste capítulo foi feito o enquadramento do *software* livre e *open source*, onde foram apresentadas algumas das suas diferenças, foram abordados os tipos de licenças e dado um enfoque sobre a utilização e impacto nas empresas tendo sido finalizado com exemplos de *software* em diversos domínios. No capítulo que se segue irá ser abordado o tema do *Cloud Computing*.

3. CLOUD COMPUTING

Com uma escalabilidade virtualmente ilimitada e com uma natureza de elevada flexibilidade, a *Cloud* tem-se tornado no parceiro fundamental para qualquer sistema que aspire a uma expansão global por forma a atingir um maior número de clientes sem os constrangimentos típicos aliados ao *hardware* e às suas especificações. Por forma a esclarecer o conceito de *Cloud Computing* o subcapítulo que se segue apresenta uma breve introdução sobre este, a explicação de alguns conceitos relacionados com a tecnologia e o devido enquadramento histórico. Seguidamente é apresentada a arquitetura sob a qual assentam os diversos tipos de *Cloud* seguido por uma explicação dos principais e mais comuns modelos de negócio implementados e disponibilizados, consequência da existência da *Cloud* e onde se termina com os diversos tipos de *Cloud*, também chamados de modelos de implementação.

3.1. Introdução

Desde que se começou a ser falado e até à atualidade, o termo *Cloud Computing* ou na sua versão abreviada, *Cloud*, é considerado um chavão, algo que está na moda, que faz notícia e atrai atenções e onde não se consegue chegar a um consenso relativamente a uma definição única, dando aso a várias interpretações (Foster et al., 2008). Em suma podemos dizer que *Cloud* é um conceito de utilização e gestão de recursos distribuídos, um modelo de negócio em que o fornecedor do serviço disponibiliza recursos, sejam eles físicos ou lógicos, à medida da utilização e necessidade do cliente com políticas de pagamento indexado a essa mesma utilização. Tendo avaliado o enorme leque disponível com definições para o termo *Cloud*, a que consideramos ser a mais adequada e que melhor enquadra o termo com a realidade é a disponibilizada pelo organismo governamental americano *NIST* (*National Institute of Standards and Technology*) e que passamos a citar (*Mell & Grance, 2011*):

"*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*"

Na sua tradução para o português podemos ler que "*Cloud Computing* é um modelo para permitir o acesso ubíquo e a pedido a um conjunto partilhado de recursos computacionais configuráveis através da rede (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser aprovisionados e disponibilizados de forma simples e célere com pouca necessidade de interação com o prestador do serviço."

O conceito em causa já é bem conhecido e pode ser feito um paralelo com o que é praticado no sector dos bens e serviços, como é o caso da água, eletricidade, ou gás natural (*Zhang, 2010*); do mesmo modo, a computação, *software*, armazenamento e serviços são disponibilizados através da Internet sem que o utilizador final necessite de saber a sua localização física ou qualquer aspeto mais técnico para deles usufruir (*Marston, 2011*), pagando consoante a sua utilização; comummente utilizada a terminologia em inglês "*pay-as-you-go*" ou "*pay-per-use*".

Este novo conceito apresenta-se como resposta à crescente procura de soluções à medida, disponibilizadas de forma rápida sem que haja necessidade de investimento em novos equipamentos, licenciamento de *software* ou formação de equipas, alargando desta forma as capacidades existentes e o tempo de resposta ao mercado, tornando-se numa mais-valia relevante para as Pequenas e Médias Empresas (PME) (*Sultan, 2011*).

O modo como a Internet está massificada e de fácil acesso fez com que fosse possível explorar e apresentar novos métodos de utilização de recursos, novos modelos de disponibilização de serviços de TIC em que uma página de Internet separa um utilizador de um determinado serviço ou aplicação que usualmente se encontrava instalada no computador mas que se encontra virtualizada num *data center* do fornecedor do serviço.

Com a maturação dos serviços de *Cloud*, da virtualização, do *software* e sistemas que o suportam, algumas empresas começam a adotar internamente este conceito, criando aquilo a que se chama *Cloud* Privada, onde alojam serviços críticos que pretendem ter total controlo sobre os mesmos, onde pode ser reutilizada a infraestrutura já disponível e controlar de certa forma questões como a segurança da informação – que em muitos casos vai contra as políticas internas das empresas por não ter o controlo e o conhecimento de onde reside a informação vital do seu negócio.

A publicação feita pelo *NIST* (*Mell, 2011*) dá-nos a definição mais abrangente, percetível e a que mais é citada. Este modelo é composto por cinco características essenciais, três modelos de serviço, ou negócio, e quatro modelos de implementação. As características que definem a *Cloud* como um modelo, e que de certa forma já foram abordadas nos parágrafos anteriores, segundo (Buyya, 2008) são: acesso aos recursos de forma autónoma e a pedido, acessos ubíquo aos recursos disponibilizados pela rede, partilha de recursos, elasticidade e escalabilidade e por último, monitorização / avaliação do serviço. Os três modelos de serviço são amplamente conhecidos pelas suas siglas e nomes em inglês, nomeadamente, *SaaS* (*Software as a Service*) ou *software* como um serviço, *PaaS* (*Platform as a Service*) ou plataforma como um serviço e *IaaS* (*Infrastructure as a Service*) ou infraestrutura como um serviço. Em termos de implementação o modelo da *Cloud* contempla a *Cloud* Privada, Comunitária, Pública e Híbrida. Ambos os modelos mencionados, tanto o de serviço como o de implementação serão abordados com maior detalhe no decorrer deste capítulo.

A literatura é extensa e dependendo do autor e o âmbito do seu estudo, é feito um enfoque e uma análise diferente de cada um dos aspectos diferenciadores e que definem a Cloud.

A perspetiva dada por (*Armbrust et al., 2009*) fornece uma visão abrangente de elementos da *Cloud* e como eles são vistos de maneira diferente pelo cliente final, aplicações e serviços disponibilizados através da Internet, ou pelo prestador do serviço, os sistema e o *hardware* que compõem o *datacenter* e tornam esta realidade possível, e em como os modelos "*... as a Service*" e em concreto o *Software as a Service* (*SaaS*), evoluíram de modo a serem comercializáveis. Os autores compreendem a *Cloud* como um agregado de tecnologia já existente, no entanto, aceitaram como novo a ilusão de que existem disponíveis recursos infinitos, o desaparecimento do compromisso inicial com recursos e um bom compromisso entre o preço / utilização a curto prazo. Com uma visão mais técnica, em parte devido a serem

estudiosos da computação em *GRID*, do inglês *Grid Computing*, os autores de (*Buyya et al., 2008*) percepcionam a computação na *Cloud* como um sistema de computação paralela e distribuída que consiste num conjunto virtualizado de computadores em que os níveis de serviço são negociados entre o consumidor e o prestador do serviço.

A visão que é apresentada por (*Borenstein & Blake, 2011*) enfatiza, para além da segurança, a questão da largura de banda e a sua importância para aplicações que requeiram o processamento e tratamento de grandes quantidades de informação, como é o caso da *Big Data*, e se se torna viável a utilização da *Cloud* ao invés do processamento local.

Cloud Computing é vista como a virtualização de aplicações, serviços e armazenamento de rápida configuração e disponibilização em que se paga conforme a utilização feita, o que para (*Deep Kaur & Chana, 2010*) em termos do custo total de propriedade se torna na maior vantagem apresentada, aliando o facto de haver uma maior redundância dos vários elementos constituintes.

Segundo (*Foley, 2008*), *Cloud Computing* é vista e explicada como o acesso a recursos virtualizados de TIC que se encontram alojados fora do seu próprio datacenter, partilhados com terceiros, simples de usar, pagos por medida e acedidos através de um computador ou qualquer dispositivo móvel, desde que, com ligação à Internet. É uma explicação de um certo modo simplista mas que descreve as características chave do conceito em causa tornando-as de fácil entendimento para o público menos ligado às novas tecnologias. Por um outro lado, (*Foster et al., 2008*) abordam o paradigma sobre o prisma de uma audiência empresarial, mais propensa às novas tecnologias, onde o descreve como computação distribuída em grande escala que é impulsionada por economias de escala, em que o poder de computação, armazenamento e serviços são disponibilizados através da Internet e dimensionados à medida das necessidades e dos pedidos de cliente externos. O impulsionamento da Cloud é potenciado pelo facto de haver um baixo custo de virtualização e de manutenção da instância de um utilizador durante o tempo acordado para o efeito.

A consultora tecnológica (*Gartner, 2009*) explica a *Cloud* como um estilo de computação onde recursos tecnológicos escaláveis e adaptativos são entregues a cliente externos "como um serviço" através da Internet. Tal como (*Armbrust et al., 2009*), ambos colocam enfoque no *SaaS*. Para a maioria dos utilizadores da Internet é esta a camada da *Cloud* com que mais interagem, derivado do uso cada vez mais generalizado das aplicações do *Google*, *Dropbox* ou *Netflix*, por outro lado, os clientes empresarias são tido como o público-alvo das camadas de plataforma (*PaaS*) ou infraestruturas (*IaaS*).

Um dos grandes parceiros no desenvolvimento da *Cloud*, membro de vários grupos para o desenvolvimento da tecnologia e um dos prestadores de serviço de enorme sucesso nas três áreas, a *IBM*, que através da *IBM SmartCloud* disponibiliza *IaaS*, *PaaS* e *SaaS*, define-a como uma solução abrangente em que todos os recursos de computação são rapidamente fornecidos aos utilizadores de acordo com as suas necessidades, de maneira simplificada e rentável de forma a suportar soluções de negócio inovadoras (*IBM, 2009*).

Como exemplo da diversidade e extensão de definições e abordagens ao tema de *Cloud Computing* foram vistos acima as opiniões de alguns autores, no entanto, a maioria das

definições são originadas pelos fornecedores de serviços, empresas de consultoria ou analistas de mercado como a *IDC* (*International Data Corporation*) (*Gens, 2008*) ou a *Gartner* (*Plummer et al., 2008*).

3.2. Desambiguação de conceitos

Quando se fala em *Cloud* é inevitável falar em *Grids*, *Clusters*, *Web 2.0* ou Computação utilitária, mas mais importante ainda, é preciso diferenciar e perceber as diferenças. Este subtópico pretende isso mesmo, dar uma abordagem breve e concisa sobre os conceitos subjacentes ao tema *Cloud Computing* que se encontram representado na Figura 3.1.

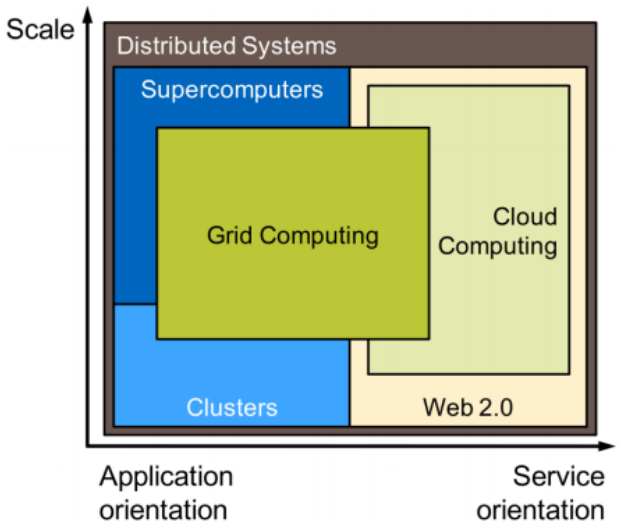

Figura 3.1 - Interligação e áreas de aplicação (*Foster et al., 2008*)

- **Supercomputadores**

Supercomputadores era a designação utilizada na década de 50, 60 e até aos inícios de 80 para designar os Mainframes, ou computadores centrais, de grande porte, bastante dispendiosos e com uma grande capacidade de processamento.

- **Grid Computing**

Grid Computing pode ser definido como o uso de recursos de computação originários de diversos domínios administrativos com as suas próprias políticas de gestão e objetivos, no entanto, com a finalidade de alcançar um propósito em comum (*Buyya, 2005*). Pode ser considerado um sistema distribuído heterogéneo em que as máquinas se encontram geograficamente dispersas ao contrário do que acontece com os *Clusters*. Os recursos disponíveis para a resolução de uma tarefa é de acordo com aquilo que cada utilizador ou entidade considera que pode disponibilizar.

- **Clusters**

Os *Cluster* diferenciam-se na sua essência por serem agregados de computação em que os recursos são homogéneos, usualmente todos dentro do mesmo domínio administrativo e administrados por uma única entidade. Também os *Clusters* agregam uma coleção de computadores, ligados em rede com a finalidade de partilha de recursos para atingir uma determinada tarefa. Os recursos disponíveis num Cluster são a soma da totalidade da capacidade de computação das máquinas envolvidas.

- **Utility Computing**

O termo em inglês *Utility Computing* é o mais comum na literatura e surge do processo de tonar disponíveis as infraestrutura e recursos de TIC do mesmo modo que é disponibilizada a eletricidade ou a água, acessíveis a qualquer um segundo um método de pagamento por utilização. O termo *Utility Computing* é muitas vezes utilizado no lugar de e/ou para caracterizar Cloud Computing, no entanto, pode ser implementado sem estar na *Cloud*. Por exemplo, um servidor que disponibiliza capacidade de processamento para a execução de tarefas de vários clientes; neste caso não existe virtualização de recursos e a localização é central.

- **Cloud Computing**

A *Cloud Computing* possui características da computação em grelha, dos *clusters* e da computação utilitária, contudo, vai mais além do que fazem e disponibilizam. Encontra-se acessível a partir da maior rede conhecida, a Internet, disponibilizando de forma virtualizada recursos e serviços (Dialogic, 2010).

- **Web 2.0**

A *Web 2.0* é um termo utilizado para designar uma segunda geração de produtos e serviços fornecidos através da Internet. O termo não se refere às especificidades técnicas mas sim como o utilizador e o programador percecionam o ambiente de novas oportunidades e desafios (*Greenhow, 2009*). A constante evolução dos conceitos, serviços assentes em tecnologia emergente leva a que estejamos no sentido de evolução para uma terceira vaga da utilização da Internet (*Gil, 2014*).

3.3. Enquadramento histórico

O conceito subjacente de *Cloud Computing* não é novo e é sem dúvida a consequência natural da evolução de tecnologias de computação. Em 1961, o cientista americano *John McCarthy* – tido por muitos como o pai da inteligência artificial – durante o discurso comemorativo do centenário do *MIT*, foi o primeiro a sugerir publicamente aquilo que hoje conhecemos como *Cloud Computing*: "S*e os computadores, da forma como eu imagino, se tornarem os computadores do futuro, então a computação poderá ser como um serviço público, assim como a água ou a eletricidade o é... Cada assinante pagará apenas pelos recursos que realmente utilizar, mas ele terá acesso a todos os recursos oferecidos pelas linguagens de programação de um grande sistema... Alguns assinantes poderão oferecer serviços a outros assinantes... A computação como um serviço público poderá ser a base de uma nova e importante indústria*".

O avanço tecnológico foi notório ao longo dos anos no que respeita às tecnologias da informação. A Figura 3.2 sumariza as etapas decorridas desde o 1º computador até à presente era, a da *Cloud Computing*.

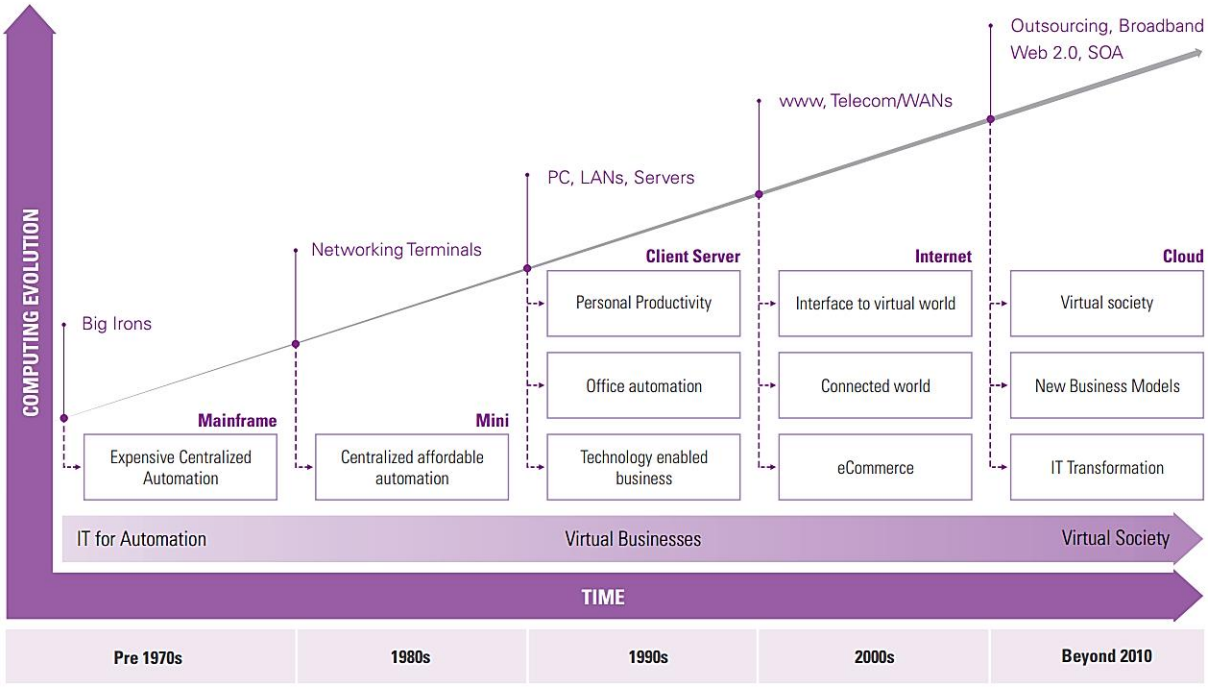

Figura 3.2 - Evolução da Computação (*KPMG, 2011*)

São vários os pontos-chave na história, no entanto, destacamos aquilo que podemos considerar como os quatro pontos de viragem no caminho traçado deste então.

- **Mainframe**

As décadas de 50, 60 e até aos finais de 70 foram marcadas pelos *Mainframes*, enormes computadores centrais responsáveis pelo processamento dos pedidos dos utilizadores. Tratava-se de uma infraestrutura bastante dispendiosa e de acesso restrito. Foi o início da automatização de tarefas.

- **Computador pessoal**

A década de 80 foi marcada pela ascensão do computador pessoal. Foi trazido para o domínio público um equipamento com capacidade de processamento e a um preço acessível. Começava a descentralização da computação e o nascimento da indústria dos serviços de TIC.

- **Cliente – Servidor**

Na década de 90 deu-se a evolução na comunicação em rede, nas redes locais, mais comummente conhecidas pelo acrónimo em inglês *LAN* (*Local Area Network*) que potenciou as arquiteturas cliente/servidor. Deu-se o arranque da *World Wide Web*.

- **Internet**

Com a Internet começou a partilha de recursos a nível mundial. As infraestruturas de TIC subiram de patamar e resultaram no que começou a ser os *datacenters*. Começam a delegar-se a terceiros a gestão e manutenção de infraestruturas. Aumento da utilização de virtualização.

- **Cloud**

Modelo da computação como uma utilidade em que surge o paradigma "*as a service*" dando vida a cada vez mais modelos de serviço.

Em 1999 surge a *Salesforce.com* como o primeiro elemento da *Cloud* onde avança com o conceito de disponibilizar *software* empresarial através de uma página da Internet. Em 2002 a *Amazon* lança o seu serviço de *Cloud*, o *Amazon Web Service* (*AWS*), seguido pela *Google* em 2006 com os seus *Google Docs*, trazendo para um domínio mais público esta realidade da *Cloud* e do *software* como um serviço. Nesse mesmo ano a *Amazon* apresenta o *Elastic Compute Cloud* (*EC2*), um serviço que permitia o aluguer de computadores virtuais. Em 2008, fruto das colaboração da *Google*, *IBM* e um enorme número de universidades americanas surge a *Eucalyptus*, a primeira plataforma *open source* compatível com *AWS* que permite a implantação de soluções de *Cloud* privadas. *OpenNebula* surge como o primeiro *software open source* para criação de soluções privadas e híbridas. Em 2009 foi a vez da *Microsoft* aparecer e apresentar o *Windows Azure*.

Desde então são inúmeras as soluções e parcerias feitas no sentido de apresentar mais, melhores, robustas e fiáveis soluções para clientes individuais e empresariais.

3.4. Arquitetura

Quando o instituto nacional norte-americano *NIST* apresentou uma definição para o que é a *Cloud Computing*, apresentou também uma referência para a sua arquitetura conceptual, identificando quais os agentes intervenientes desse ecossistema que estava a emergir.

Embora para o consumidor final o que transparece do modelo de *Cloud* é o serviço prestado, existem agentes ou entidades envolvidas que complementam o modelo e interagem na prestação do serviço pretendido. Nesse sentido, este subcapítulo pretende dar ao leitor uma visão de alto nível dos elementos e funções de cada um na arquitetura do modelo de Cloud. Num primeiro plano encontram-se cinco elementos, conforme ilustrado na Figura 3.3.

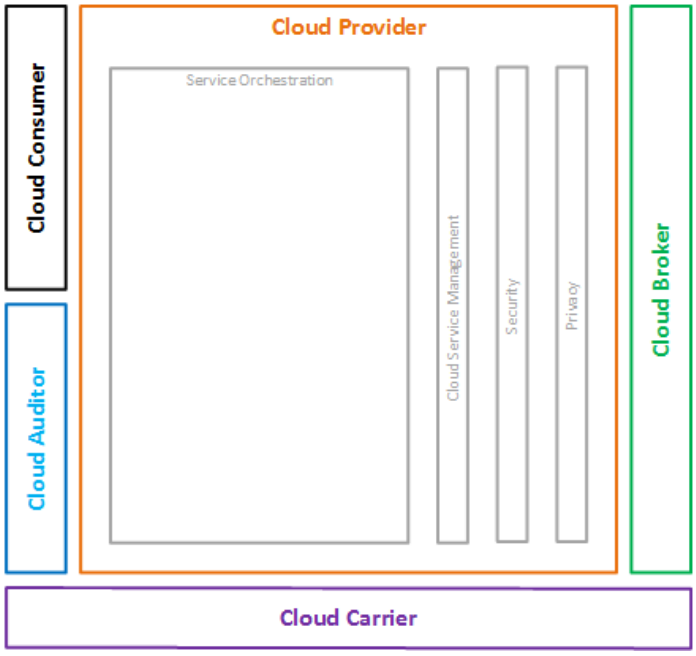

Figura 3.3 - Modelo conceptual de referência para a arquitetura da *Cloud* (*Kreger, 2012*)

- **Consumidor ou Cliente (*Cloud Consumer*)**

O utilizador final pode ser um indivíduo ou uma organização que adquirem ou utilizam produtos e serviços de *Cloud*.

- **Fornecedor de Serviços (*Cloud Provider*)**

O fornecedor é a entidade que em última instância fornece o serviço, nomeadamente, serviços de *software*, infraestruturas ou plataformas. Ao fornecedor do serviço compete a garantia da segurança e privacidade dos dados do cliente, bem como o cumprimento dos níveis de serviço (*SLA*) que foram definidos aquando da contratualização do serviço.

- **Intermediário (*Cloud Broker*)**

O intermediário atua com um intermediário entre os utilizadores finais e fornecedores de serviços. É um elemento facilitador junto dos consumidores desmistificando a complexidade das ofertas de serviços *Cloud*, podendo também criar serviços *Cloud* com valor acrescentado.

- **Auditor (*Cloud Auditor*)**

O auditor é um organismo independente que pode realizar avaliações de controlo dos serviços de *Cloud* com a intenção de expressar uma mesma opinião. As auditorias são realizadas para verificar a conformidade com as normas através de uma revisão de provas objetivas. Um auditor pode avaliar os serviços fornecidos no que respeita às medidas de segurança implementadas, impacto sobre a privacidade ou níveis de desempenho.

- **Transportador (*Cloud Carrier*)**

O *Cloud Carrier* é a entidade que assegura o canal de comunicação entre os fornecedores e os consumidores de serviços *Cloud*, que tipicamente são os operadores de infraestruturas de telecomunicações. Estas entidades podem eventualmente acumular funções de fornecedores de serviço.

Se, a partir do modelo apresentado anteriormente na Figura 3.3, for diminuído o nível de abstração e nos focarmos no fornecedor de serviços, temos o núcleo de um qualquer serviço de *Cloud Computing* que seja apresentado ao cliente final. A Figura 3.4 apresenta os componentes que constituem esse núcleo.

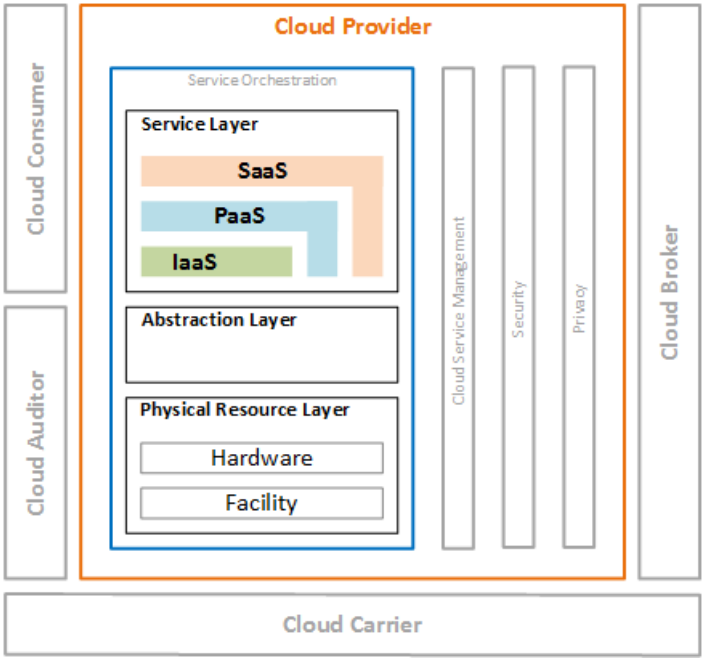

Figura 3.4 - Arquitetura dos serviços de *Cloud* (*Kreger, 2012*)

- **Camada física (*Physical Resource Layer*)**

Esta camada, considerada com a espinha dorsal de um sistema de *Cloud*, representa toda a estrutura física necessária para a sua implantação, desde o espaço físico, sistemas de refrigeração, servidores ou sistemas de comunicação como *routers* e *switchs*. É ao nível desta camada que é feita a configuração do *hardware* de acordo com a solução a implementar, onde são configurados os sistemas de tolerância a falhas e configurada a rede.

- **Camada de abstracção (*Abstraction Layer*)**

Esta camada é responsável por providenciar as capacidades básicas de virtualização e gestão dos recursos físicos da camada que se encontra abaixo, disponibilizando desta forma recursos sem que o utilizador necessite de se preocupar com o que existe ou está configurado. Este nível de abstração é conseguido com recurso aos hipervisores (*hypervisors*), gestores de máquinas virtuais ou de armazenamento. Camada onde é feita a dissociação entre os recursos físicos e os recursos virtuais, por exemplo, no nível anterior poderão ter sido configurados discos rígidos em sistema *RAID* e o que transparece para cima é uma determinada capacidade de armazenamento que será utilizada na solução a implementar.

- **Serviços (*Service Layer*)**

A camada superior, o topo da pirâmide, é chamada de camada de serviço, pois é neste nível que são definidos os interfaces de acesso de cada um dos três modelos de serviço disponibilizados por uma solução de *Cloud Computing*, nomeadamente, *IaaS*, *PaaS* ou *SaaS*. Embora os três modelos sejam representados graficamente como interdependentes, visto que uma solução de *PaaS* pode ser construída em cima dos módulos já existentes da *IaaS* e por sua vez uma solução de *SaaS* pode assentar em cima da camada anterior, esta sucessão não é necessariamente um requisito obrigatório. Por exemplo, uma solução de *Software as a Service* pode ser implementada dentro de uma máquina virtual criada numa infraestrutura de *Cloud* como pode ser disponibilizada diretamente a partir de recursos virtualizados da infraestrutura.

3.5. Modelos de Serviço ou Negócio

Conforme referido inicialmente, *Cloud Computing* é baseado num conjunto de tecnologias já existentes e bem estudadas, como por exemplo, computação distribuída, *Grid Computing* ou virtualização. Embora muitos dos conceitos não sejam recentes, a grande inovação da *Cloud* assenta no modo como os serviços de computação são oferecidos.

De uma forma muito simplista o fator diferenciador de cada um dos modelos identificados está no público-alvo e na quantidade de funcionalidades disponíveis, aumentando ou diminuindo por consequência o grau de responsabilidade ou conhecimento exigido a esse mesmo público. A Figura 3.5 é representativa disso mesmo.

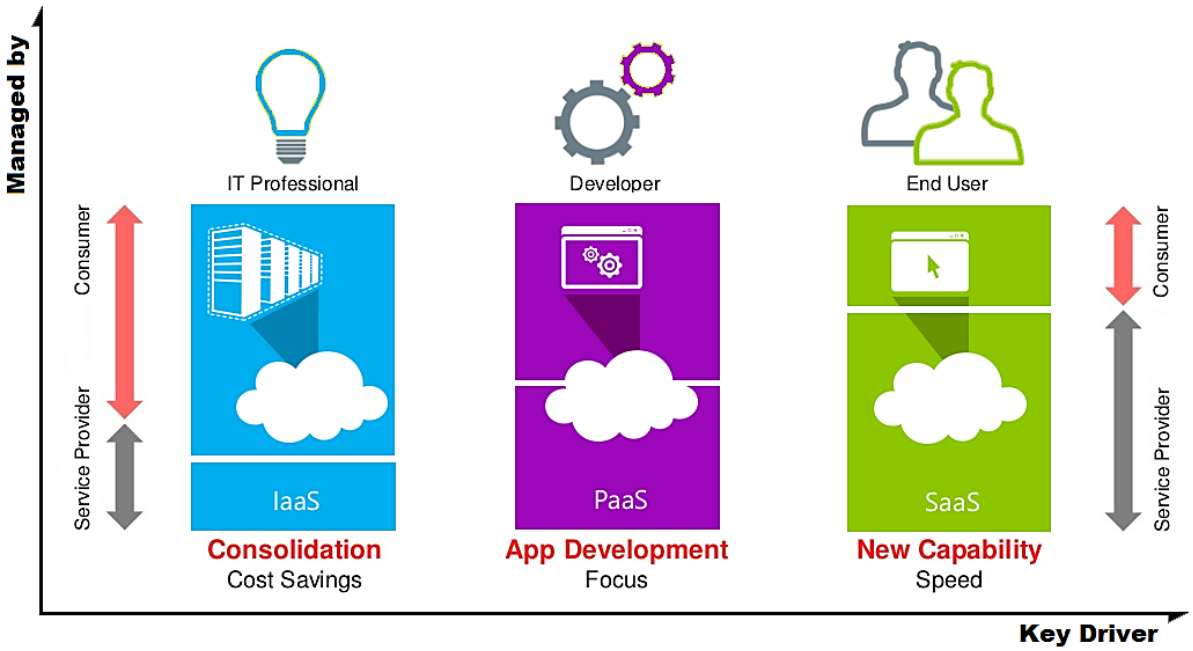

Figura 3.5 - Modelos de Serviço ou Negócio (*What is a Public Cloud, 2015*)

Embora os modelos de negócio de um modo geral e abrangente sejam três, *Infrastructure as a Service* (*IaaS*), *Platform as a Service* (*PaaS*) e *Software as a Service* (*SaaS*), tem havido uma constante evolução no que é disponibilizado ao cliente segundo a premissa de serviço ou "*as a service*", como é o caso de Comunicações (*CaaS*), Bases de Dados (*dBaaS*) ou monitorização (*MaaS*). Devido à diversidade da oferta começa a ser usual encontrar na literatura o termo *XaaS* ("*anything as a service*" ou "*everything as a service*") que remete para "tudo como um serviço" que nada mais é do que uma especialização do modelo *SaaS*.

De seguida são explicados cada um dos três modelos de prestação de serviços de Cloud.

3.5.1.IaaS – Infraestrutura como um Serviço

A camada de infraestrutura é o alicerce da pirâmide do ambiente de *Cloud*, sobre o qual assentam os serviços de plataformas e *software*. Ao disponibilizar uma Infraestrutura como um Serviço (*Infraestructure as a Service - IaaS*): é a maneira que o fornecedor tem de oferecer capacidade de processamento e armazenamento de forma transparente. Neste cenário, o utilizador final não tem a gestão da infraestrutura física mas, através de mecanismos de virtualização e máquinas virtuais, possui o controlo sobre os sistemas operativos, volumes de armazenamento, aplicações instaladas e, possivelmente, um controle limitado dos recursos de rede, em que os fornecedores geralmente cobram tais serviços com base na utilização ou configuração e numa base horária ou às vezes mensal.

Dentro deste nível podemos ter segmentos que oferecem soluções de computação, de armazenamento ou comunicações disponibilizadas por entidades conhecidas, tais como: "*Amazon Elastic Cloud (EC2)*", "*JitScale*", "*Rackspace*", "*Openstack*", "*Uniserver*", "*EMC*" ou "*Symantec*".

É sobre esta camada da arquitetura de serviços Cloud que se insere o trabalho aqui apresentado. Foi estudada e avaliada uma plataforma *open source* que irá assentar sobre hardware já existente numa empresa a fim de disponibilizar um conjunto de máquinas virtuais com a finalidade de reutilizar e rentabilizar equipamentos.

Os pontos fortes e fracos ou vantagens e pontos menos favoráveis para se optar por uma solução de *IaaS*, irão ser sempre determinados mediante o modelo implementado, público ou privado, no entanto, em termos gerais, a Tabela 3.1 resume esses mesmos.

Tabela 3.1 - Pontos fortes e menos favoráveis na adoção de uma solução de *IaaS* (*Sandoval, 2015*)

Pontos Fortes	Pontos menos favoráveis
Redução de custos operacionais Maior controlo sobre a infraestrutura Ambiente não tão restritivo	Dependente de ligação à rede Valores de serviço algo elevados

3.5.2.PaaS – Plataforma como um serviço

Plataforma como um Serviço (*Plataform as a Service - PaaS*) tem por objetivo facilitar o desenvolvimento de aplicações criando uma plataforma que agiliza esse processo ao libertar o programador das questões relacionadas com a configuração e escalonamento da infraestrutura. O *PaaS* oferece uma infraestrutura de alto nível de integração para implementar e testar aplicações em que é composto por todos os recursos necessários para o desenvolvimento de *software* em uma ou mais linguagens de programação tais como compiladores, bibliotecas, base de dados e um sistema operativo.

De notar que, o ambiente de desenvolvimento pode ter limitações quanto às linguagens de programação, bases de dados ou sistema operativo, ou seja, não é uma plataforma na sua globalidade genérica, mas sim uma plataforma completa para uma determinada finalidade. Também é comum ser utilizado para o alojamento de *sites* ou disponibilizar soluções de *software*.

Ao subir na hierarquia das soluções de *Cloud* é natural que diminua a responsabilidade de gestão e manutenção da infraestrutura de suporte, ficando isso a cargo do fornecedor do serviço, no entanto, em contrapartida, diminui o nível de flexibilidade em termos de configurações personalizadas e à medida tanto quanto poderiam ser desejáveis.

Dentro deste nível podemos ter segmentos que oferecem plataformas para desenvolvimento específico, como é o caso da "*Salesforce.com*" ou "*SAP Business ByDesign*" ou plataformas mais genéricas onde se encontram igualmente grandes nomes como a "*Google App Engine*" ou a "*Microsoft Azure*".

A Tabela 3.2 descreve, em termos gerais, os pontos fortes e fracos ou vantagens e pontos menos favoráveis ao adotar uma solução de *PaaS*.

Tabela 3.2 - Pontos fortes e menos favoráveis na adoção de uma solução de *PaaS* (*Sandoval, 2015*)

Pontos Fortes	Pontos menos favoráveis
Maior facilidade de migrar uma aplicação para a *Cloud* Maior controlo relativamente a uma solução aplicacional *SaaS*	Limitado às especificações da plataforma Maior dependência do fornecedor da infraestrutura

3.5.3.SaaS – Software como um serviço

Na ótica do utilizador final, a camada de topo, que é conhecida neste modelo como *Software* como um Serviço (*Software as a Service – SaaS*) é a que tem maior visibilidade. A este nível é onde existe o maior grau de abstração face às camadas inferiores pois o utilizador não sabe, nem necessita de saber, quais os recursos físicos, linguagens de programação ou mesmo o sistema operativo.

É normal que este tipo de serviços seja uma conjugação de outros serviços alojados na *Cloud* contudo, a interação com o utilizador final é feita por um único ponto de entrada, usualmente uma página de Internet.

A adoção cada vez maior destes serviços em ambiente empresarial das empresas, em parte deve-se a fatores como a redução de custos operacionais com a manutenção de sistemas físicos e lógicos, licenciamento de *software* ou correções evolutivas que ficam a cargo do prestador, por outro lado levantam-se questões como a segurança dos dados e disponibilidade do sistema.

Dentro deste nível, das soluções aplicacionais alojadas na nuvem, existe uma maior segmentação e oferta, como por exemplo em termos de Comunicação e Cooperação existem soluções como o "*Cisco Webex*" ou o "*Microsoft Skype for business*", em termos de Produtividade e *Office* encontramos o "*Microsoft Office 365*" ou o "*Google Apps*", soluções CRM enquadram-se as ofertas "*Salesforce.com*" e "*PerfectView CRM Online*" ou para o setor do retalho o "*Descartes*" ou "*JDA Software*".

A Tabela 3.3 descreve, em termos gerais, os pontos fortes e fracos ou vantagens e pontos menos favoráveis ao adotar uma solução de *SaaS*.

Tabela 3.3 - Pontos fortes e menos favoráveis na adoção de uma solução de *SaaS* (*Sandoval, 2015*)

Pontos Fortes	Pontos menos favoráveis
Muitas soluções gratuitas Fácil utilização / acesso Maior escolha / oferta diversificada	Não existe controlo sobre "nada" Maior oferta maior competição para fazer singrar uma determinada solução

3.6. Modelos de Implementação

Independentemente das classes de serviços até agora explicadas, atualmente as soluções de *Cloud* podem ser implementadas ou classificadas segundo três tipos principais, denominadas de *Cloud* Pública, Privada ou *Cloud* Híbrida. A grande diferença entre elas encontra-se na localização de todos os componentes que constituem a solução, ou seja, se a infraestrutura está localizada fora do domínio operacional da empresa, totalmente dentro das instalações da empresa, chamada de *On-Premise*, ou se é um misto das duas.

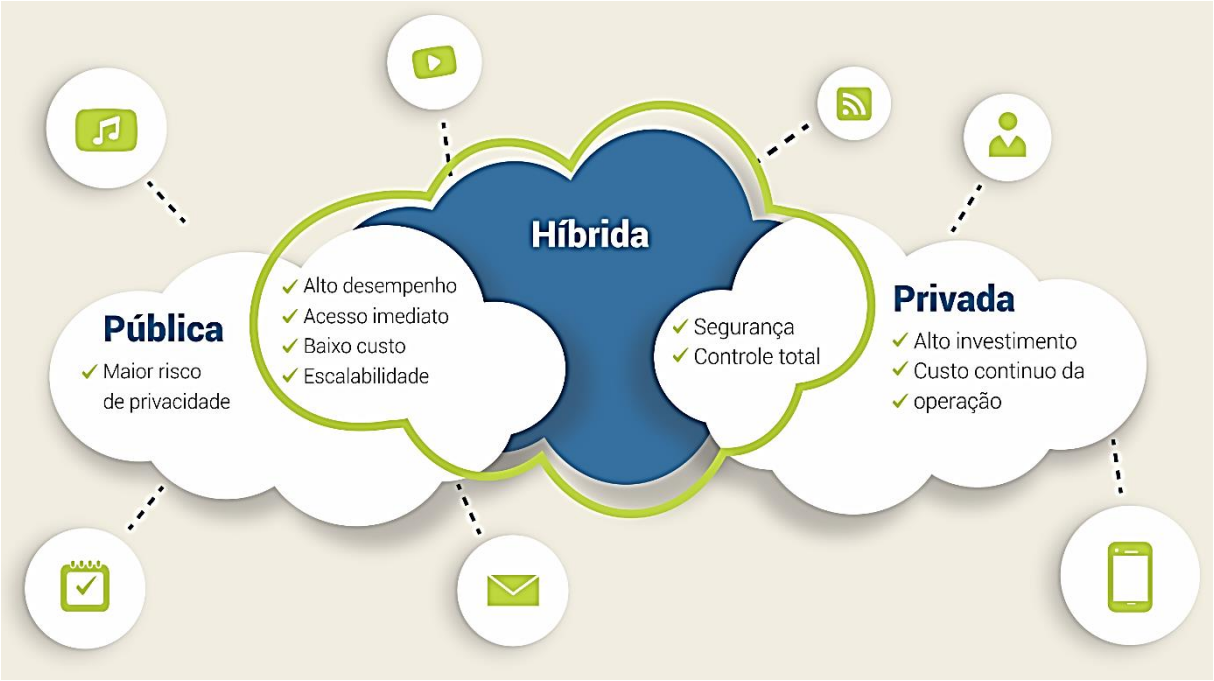

Figura 3.6 - Modelos de implementação de uma solução de *Cloud Computing* (*Cloud Computing, 2016*)

A adoção de uma ou mais em detrimento de outra será determinada pela necessidade, finalidade ou objetivo primário da empresa. De seguida será dada uma breve explicação do âmbito de cada uma das soluções apresentadas.

3.6.1. Cloud Pública

Neste caso, quando se fala que é uma Cloud Pública não deve ser interpretado literalmente - embora existam algumas ofertas livres de custos, na sua generalidade significa que a sua utilização pode ser feita por qualquer entidade desde que sejam respeitadas as regras de utilização e pagamento - não significa que a informação esteja acessível por todos, muito pelo contrário, são implementadas cada vez mais, maiores medidas de controlo de acessos para garantir a confiabilidade do serviço e a fidelização de novos clientes (Carissimi, 2015).

A oferta das nuvens públicas assenta num conjunto de recursos e poder de computação disponibilizada através da Internet onde os clientes são autónomos para escolher e configurar o que pretendem, sendo que o pagamento, por norma, é indexado à utilização. No momento da contratualização deste tipo de serviço junto da entidade detentora do mesmo é usual serem especificados os níveis de serviços (*Service-level Agreement – SLA*) e de qualidade de serviço (*Quality of servisse – QoS*) a serem cumpridos.

Vantagens

- Investimento inicial nulo pois a construção da infraestrutura é da responsabilidade do fornecedor da solução;
- Parte do risco em termos de gestão e manutenção da infraestrutura é do fornecedor;
- Capacidade de resposta, poder de computação e armazenamento, é usualmente maior face às *Cloud* privadas, facilitando previsões de crescimento ou até mesmo o crescimento repentino;
- Em termos empresariais o capital destinado ao *Capex* pode ser orientado para *Opex* (*Maverick, 2015*).

Desvantagens

- O controlo que o cliente tem relativamente à infraestrutura, segurança e rede é usualmente pouca ou nenhuma;
- Dependentes das especificações do fornecedor o que pode tornar o serviço menos flexível face a uma solução privada;
- Flutuação dos níveis de serviço assegurados pelos diversos fornecedores;
- Propriedade legal de dados e informações de privacidade muitas vezes não se encontra bem documentada.

3.6.2.Cloud Privada

As soluções de *Cloud* Privada, ao contrário do modelo anterior em que o ambiente aloja vários clientes, assentam na premissa de que a infraestrutura é utilizada exclusivamente por uma única pessoa ou organização em que os recursos físicos são propriedade do cliente e usualmente instalados dentro do domínio operacional deste, no entanto, consoante os casos, os clientes ou necessidades específicas, acontece terem o *datacenter* externo à empresa tal como entregarem a gestão e manutenção a terceiros, num sistema de *outsourcing* (Carissimi, 2015).

Esta é uma solução disruptiva face ao modelo de negócio clássico em que o utilizador paga por utilização, pois todos os custos e riscos associados são da inteira responsabilidade do cliente.

Vantagens

- O cliente tem o controlo total, literalmente, sobre a infraestrutura; o que por outro prisma pode ser visto como uma desvantagem se não possuir conhecimento técnico especializado para a gestão e manutenção.
- Maior liberdade, seja em termos dos recursos disponíveis como configurações ou teste de novas tecnologias.
- Pode tornar-se num agente facilitador na migração para ambientes virtualizados, na passagem de um ambiente tradicional para a migração do negócio ou parte dele para a *Cloud*.

Desvantagens

- Enorme investimento inicial, contudo, acaba por ser melhor no que concerne à relação custo/eficiência.
- Requer bastante tempo para colocar em perfeito funcionamento, com as garantias de segurança, monitorização e gestão de risco.

O estudo feito e apresentado nesta dissertação contempla uma nuvem privada, com recurso a *software open source*, de modo a rentabilizar *hardware* disponível dentro da empresa e oferecer internamente melhores e mais rápidas soluções face ao TIC tradicional.

3.6.3.Cloud Híbrida

Parte Cloud pública, parte Cloud privada, este modelo de Cloud híbrida não é mais que uma implementação do que há de melhor de uma ou mais nuvens de outros tipos ligadas entre si. (*CSCC, 2016*)

Tipicamente, neste modelo, os utilizadores usufruem da Cloud pública para alojar serviços não críticos da empresa enquanto mantêm o *core* dos serviços do negócio alojados internamente. Este tipo de abordagem de nuvem híbrida pode funcionar bem para empresas que de outra forma não podem usar uma nuvem pública por razões de segurança ou regulamentares.

Vantagens

- Maior flexibilidade visto juntar o melhor de dois mundos, *Cloud* pública e privada;
- Maior controlo face às nuvens públicas e escalabilidade mais facilitada em comparação com as privadas;
- Maior isolamento de certas áreas de negócio face a uma solução totalmente pública.

Desvantagens

- Complexidade adicional na interligação entre ambos os ambientes;
- Análise cuidada de qual a informação e ou área de negócio que fica em qual modelo e a interligação necessária entre si.

3.6.4.Cloud Comunitária

As plataformas de *Cloud* comunitárias possuem uma infraestrutura física própria, em que esta é detida, gerida e partilhada por organismos que possuem um mesmo objetivo e que, normalmente estão sujeitos a um mesmo tipo de restrição legal ou de segurança. Um exemplo deste tipo de implementação pode ser o caso de uma Universidade em que os recursos de computação das diversas faculdades podem ser partilhados para o bem comum.

Vantagens

- Os custos poderão ser menores devido à partilha dos mesmos entre as entidades.

Desvantagens

- Dificuldade na gestão das restrições de segurança específicas de cada entidade;

- Maior dificuldade de gestão e manutenção devido a serem ambientes partilhados.

Neste capítulo foi apresentado o *Cloud Computing*, ou na sua tradução para o português, computação na nuvem. Foi dada uma breve explicação do conceito bem como as várias perspetivas de diversos autores seguido da explicação dos termos e conceitos inerente à tecnologia, complementando com um resumo da evolução tecnológica até atingir o patamar atual onde se encontra inserido este tema. Foi apresentada a arquitetura, os seus modelos de serviço ou negócio e concluído este capítulo mostrando quais os modelos de implementação disponíveis para o mercado. No capítulo que se segue irão ser apresentadas várias plataformas de *Cloud Computing* onde no final será feita uma breve comparação entre estas.

4. PLATAFORMAS CLOUD COMPUTING OPEN SOURCE

No capítulo anterior foi dado uma introdução à computação na nuvem, mais comummente chamada de *Cloud* ou *Cloud Computing*, onde foi explicado a sua essência e características, apresentando a evolução histórica, arquitetura, os seus modelos de negócio e de implementação.

A ideia principal base da *Cloud Computing* é a mudança do forte investimento e da complexidade inerente às novas tecnologias para o lado dos fornecedores, orientando o foco para o reinvestimento na aquisição de serviços.

O fenómeno crescente de soluções de *Cloud Computing* privadas implementadas em empresas de menor escala como as *PME* pode ser entendido pelo rápido crescimento do conceito de *Cloud* conjuntamente com o amadurecimento das soluções de virtualização, e que, desde que bem pensadas e implementadas, podem de facto ser uma mais-valia para quem as coloca em prática, seja como forma de sustentar novos projetos ou como desenvolvimento de competências internas. De seguida são apresentados os projetos *CloudStack*, *Eucalyptus*, *Nimbus*, *OpenNebula* e *OpenStack*, que foram tidos em consideração devido à sua forte presença em toda a literatura estudada no âmbito deste trabalho, que visa uma solução *open source* para construção de uma infraestrutura de *Cloud Computing* privada.

4.1. CloudStack

O aparecimento deste projeto remonta ao ano de 2008 pela mão da empresa *startup* de nome VMOps que algum tempo depois alterou o nome para Cloud.com. Foi sob esse nome que, em 2010, disponibilizou parte do código fonte do CloudStack com a licença GPLv3. Em 2011 foi comprada pela Citrix e passado um ano doada à Apache (Shapeblue, 2016), a qual mais tarde lançou a primeira versão estável da plataforma de Cloud (CloudStack, 2016).

Embora não tenha uma integração tão profunda e quase exclusiva para os serviços da *Amazon* como é o caso da plataforma *Eucalyptus*, esta tem integração com o *EC2* e *S3* e no caso de armazenamento é possível integrar o componente *Swift* do *Openstack*. Embora com filosofias diferentes no que concerne aos seus modelos de arquitetura e componentes, ambas as plataformas beneficiaram de um denominador comum, a Citrix, que impulsionou o desenvolvimento de ambas as plataformas.

A arquitetura desta plataforma de Cloud é bastante simples em termos do número de elementos que a constituem conforme se pode observar na Figura 4.1, contudo, é possível criar infraestruturas de Cloud complexas e fisicamente distribuídas.

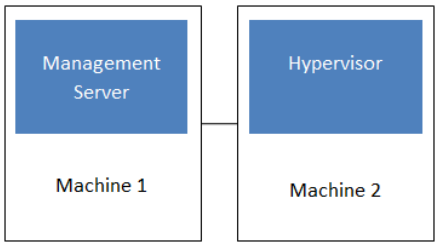

Figura 4.1 - Arquitetura de uma instalação básica (*Deployment Architecture Overview, 2016*)

De seguida são apresentados os elementos que constituem a arquitetura da plataforma CloudStack.

Servidor de Gestão: Este elemento, como o próprio nome indica é responsável pelas tarefas de gestão da plataforma de *Cloud*, de todos os recursos que constituem a infraestrutura tais como máquinas físicas ou *hypervisors*, armazenamento, dispositivos e endereçamento *IP*. É este servidor que faz a alocação dos pedidos de máquinas virtuais, atribuição de *IP* internos e externos para comunicação com a Internet, atribuição de espaço de armazenamento e gestão das imagens para as máquinas virtuais entre outras ações. A gestão da infraestrutura de *Cloud* é feita através de um interface *web* que é suportado pelo servidor *web* da *Apache, Tomcat,* e a informação guardada numa base de dados *MySQL*.

Hypervisor: Este elemento da infraestrutura é responsável por ter o *hypervisor* instalado e pronto a executar as máquinas virtuais bem como fornecer os recursos de computação.

Estes mesmos recursos, mediante agregação de mais elementos e/ou separação em segmentos diferentes de rede, dão origem a soluções mais complexas desde *Clusters*, *Pod* – um *cluster* ou mais desde que fisicamente dentro do mesmo endereçamento de rede – até Regiões – conjunto de recursos de computação que se encontram separados geograficamente.

Em termos do tipo de suportes, o *CloudStack* é bastante flexível. A sua principal linguagem de programação é *Java*, porém existem clientes em *Python*, *PHP* e *Perl*. Em termos de sistemas operativos tem suporte para *Linux* nas distribuições *RHEL*, *CentOS* e *Ubuntu* e em que utiliza os *hypervisors Xen*, *KVM*, *VMWare* e *Hyper-V*. Tal como o *OpenNebula* utiliza como principal sistema de ficheiros o *standard NFS*, o que acarreta tarefas mais específicas de configuração por parte dos administradores de sistemas mas também tem suporte para o *Swift* do *OpenStack*.

4.2. Eucalyptus

Eucalyptus (*Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems*) (*Eucalyptus, 2016*) teve as suas raízes no projeto de investigação conhecido por *VGrADS* (*Virtual Grid Application Development Software*) (*The VGrADS Project, 2016*) levado a cabo por diversas universidades americanas entre os anos de 2003 e 2008, quando, a empresa que deu o nome a este novo projeto, chefiado pelas mãos de *Rich Wolski* da Universidade de Santa Barbara na Califórnia, lançava a plataforma hoje conhecida como *Eucalyptus*. Esta plataforma surgiu da necessidade de interligar o projeto *VGrADS* com a solução de *Cloud* pública da *Amazon*, o que mais tarde, em 2012, se veio a tornar numa parceria oficial através da assinatura de um acordo formal (*Nurmi, 2009*). Mais recentemente, em 2014, foi adquirida pela *Hewlett-Packard*.

A decisão inicial por parte da empresa de interligar os seus projetos com a *Cloud* da *Amazon* orientou o desenvolvimento e tornou-a na maior plataforma que melhor se integra, apresentando a mais vasta compatibilidade face às restantes infraestruturas *open source*. Atualmente é compatível com a *Amazon EC2*, o serviço de armazenamento *Amazon S3* ou o serviço de autenticação *Amazon IAM* (*Official, 2016*).

A solução é composta por vários componentes de *software* que devidamente interligados fornecem as funcionalidades de uma infraestrutura de *Cloud* e que em seguida, é dada uma

visão geral do papel que cada um desempenha na solução final. Esses vários componentes estão ilustrados na Figura 4.2, representativa da sua arquitetura conceptual.

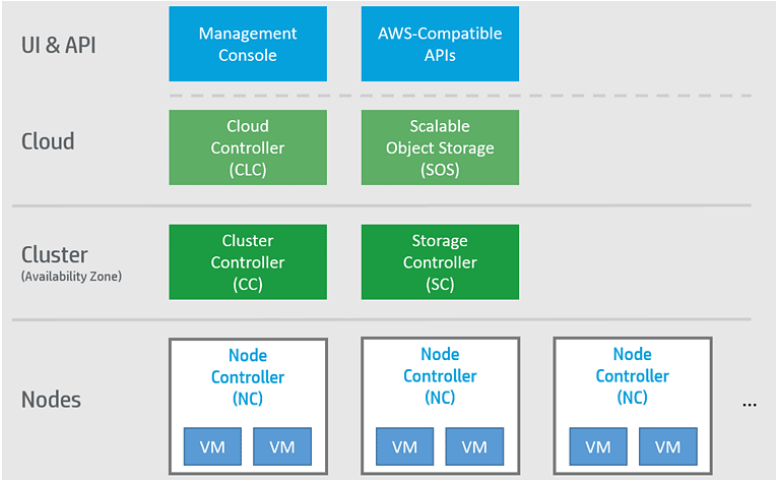

Figura 4.2 - Arquitetura conceptual da solução de *Cloud Eucalyptus* (*HPE, 2015*)

User-Facing Services (UFS): É o nome dado aos *web services* implementados internamente e que por sua vez também servem de ponte para a plataforma de serviços da *Amazon*. Ou seja, os utilizadores interagem com o sistema usando as mesmas ferramentas e interfaces que usam para interagir com a *Amazon Elastic Compute Cloud* (*EC2*).

Management Console: É o interface dado ao utilizador, sob a forma de uma página de Internet, que lhe permite interagir com a infraestrutura de *Cloud*, seja ele um utilizador final ou administrador da mesma.

Scalable Object Storage: É o nome utilizado para descrever o sistema de armazenamento, seja sob a forma de imagens das instâncias ou blocos de espaço para agregar às máquinas virtuais, que em conjunto com outros interfaces implementados, disponibilizam um serviço compatível com a *Amazon S3* (*Amazon Simple Storage Service*). Para o efeito, estão disponíveis o *Walrus*, comummente utilizado em soluções de complexidade mais baixa, o *Riak CS* e o *Ceph-RGW*.

Cloud Controller (CLC): É o módulo, criado em *Java*, que gere a base de dados com toda a informação referente à infraestrutura de *Cloud* criada. Por cada infraestrutura criada só pode haver um *CLC*. Na versão mais recente também é responsável pelo *DNS* interno de comunicação entre os serviços da *Cloud*.

Cluster Controller (CC): É o módulo, desenvolvido em linguagem *C*, responsável por saber quais as máquinas físicas disponíveis dentro da infraestrutura de modo a conseguir responder a pedidos de execução de máquinas virtuais. É também responsável pela gestão da rede utilizada para comunicação entre máquinas virtuais e serviços da plataforma. Cada solução de *Cloud* pode ter vários *Clusters*, em que para cada um tem de ser criado um *SC* (*Storage Controller*) e um *NC* (*Node Controller*).

Storage Controller (SC): Módulo, escrito em *Java*, que disponibiliza funcionalidades semelhantes à *Amazon EBS*. É o interface para vários sistemas de armazenamento que por sua vez podem ser ligados a máquinas virtuais.

Node Controller (NC): É o módulo, escrito em linguagem *C*, que controla o equipamento físico onde as máquinas virtuais irão ser executadas. É responsável pela comunicação entre o sistema operativo local e o *hypervisor*.

A solução de *Cloud Eucalyptus*, atualmente detida pela *HP* (*Hewlett Packard*), foi contruída em *Java* e para além desta linguagem *open source*, também suporta o *PHP*. Em termos de sistemas operativos o suporte contempla somente as versões de *64bits* das distribuições de *Linux CentOS 7* e o *Red Hat Enterprise Linux 7*, já no campo dos *hypervisors* está contemplado o suporte para o *KVM*, o *Xen* e o *VMWare*.

4.3. Nimbus

Tendo sido apresentado oficialmente em 2009, este projeto viu o seu crescimento ser fruto de investigação científica (Nimbus, 2015). Desde então que mantem uma estreita relação com a comunidade, ao ponto de se apresentar como sendo um conjunto de ferramentas focado em disponibilizar infraestruturas de *Cloud* para a comunidade científica. Este é um projeto com os objetivos bem definidos que se apresenta de forma simples e clara.

A solução *Nimbus* consiste um dois produtos, nomeadamente, Plataforma e Infraestrutura.

Plataforma: É um conjunto integrado de ferramentas *open source* que permitem ao utilizador tirar o máximo proveito de soluções de *IaaS*, automatizando tarefas e simplificando a gestão e monitorização dos recursos da *Cloud*

Infraestrutura: É um conjunto de ferramentas, também elas *open source*, que permitem criar as infraestruturas de *Cloud* em que a missão é primeiramente dar ênfase nas necessidade da ciência.

Ao contrário do que acontece com as plataformas anteriores, em que há um módulo específico que é necessário ser instalado para que seja possível a comunicação e gestão de rede, comunicação interna entre serviços, *VM's* e passagem para o exterior, com o Nimbus basta que cada nó tenha o componente *SSH* instalado e acesso a um servidor de *DHCP* para que a comunicação necessária entre os diversos componentes se estabeleça; no entanto, torna esta abordagem menos flexível. A Figura 4.3 é ilustrativa desses mesmos componentes, do nó principal, e de cada um dos nós que fornecem os recursos físicos necessários para a infraestrutura de *Cloud*.

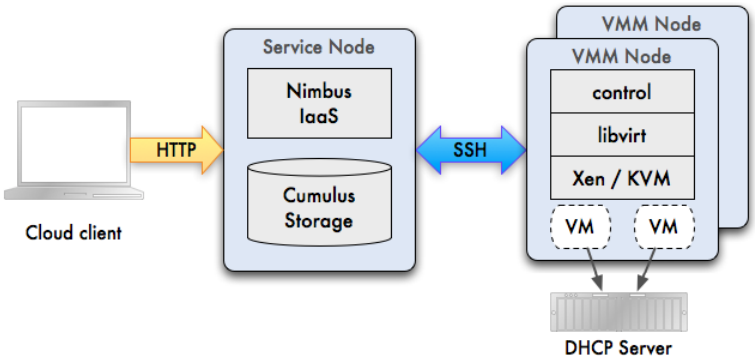

Figura 4.3 - Principais componentes da arquitetura do *Nimbus* (*Zero to Cloud Guide, 2016*)

Em termos de linguagens de programação suporta o *Java* e o *Python*, por outro lado, como tem por base muitos componentes *standard*, não há objeção quanto às versões de *Linux* em que

pode ser instalado. *Nimbus* suporta certificados *x509* para a autenticação, tem como *hypervisors* o *KVM* ou o *Xen* e como elemento fulcral – do mesmo modo que o *Swift* para o *OpenStack* ou o *Walrus* para o *Eucalyptus* – o *Cumulus*, entidade responsável pela gestão do espaço de armazenamento das imagens utilizadas para criar máquinas virtuais. Tal como acontece com outras soluções de infraestruturas de *Cloud*, também a *Nimbus* é compatível com os serviços *EC2* e *S3* da *Amazon*.

4.4. OpenNebula

Tudo começou no ano de 2005 como um projeto de investigação levado a cabo pelas mãos de *Ignacio M. Llorente* e *Rubén S. Montero*, no entanto, só foi lançado ao público em 2008, ano esse e seguintes, bastante férteis em soluções de *Cloud* que viram a luz do dia, como por exemplo, o *Eucalyptus*, *OpenStack*, *Nimbus* ou o *CloudStack*. A plataforma *OpenNebula* é o resultado de anos de pesquisa e desenvolvimento na área de virtualização em grande escala em infraestruturas distribuídas onde contou com o financiamento e a participação de vários projetos de investigação europeus (*OpenNebula, 2015*).

O principal foco desta solução de *Cloud* é a virtualização de *datacenters* e os seus recursos físicos em nuvens privadas, podendo ser adaptado a configurações simples até infraestruturas físicas geograficamente separadas. Ao contrário do que acontece, por exemplo, com o *OpenStack*, a apresentação do *software* para a construção da plataforma é feita sob a forma de único módulo ou pacote que, segundo a comunidade, é de fácil atualização para uma versão superior.

O *OpenNebula* assume que se dispõe de uma estrutura clássica de *datacenter*, nomeadamente, um ponto de acesso para gestão da infraestrutura, sistema de armazenamento de dados, capacidade de computação para execução de máquinas virtuais e uma rede que interligue todos estes elementos. A Figura 4.4 é exemplo disso mesmo.

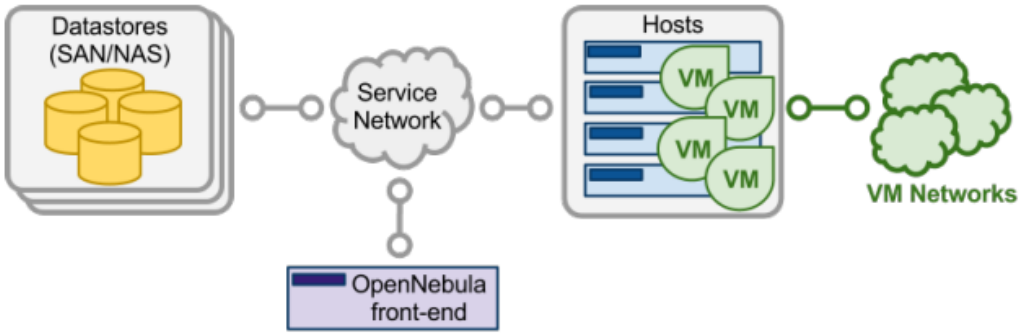

Figura 4.4 - Visão de alto nível da arquitetura (*Open Cloud Architecture, 2016*)

Em termos conceptuais o *OpenNebula* divide-se em três categorias, grandes grupos ou camadas, nomeadamente, *Drivers*, *Interface* e *Tools* (Ferramentas), em que cada uma define os serviços que capacitam esta solução para se integrar com outras e com o exterior, conferindo-lhe assim a flexibilidade enunciada pelo projeto (*Montero, 2012*). A Figura 4.5 é ilustrativa das camadas de serviços da solução.

O *OpenNebula* apresenta uma arquitetura altamente modular que oferece suporte para uma vasta gama de soluções existentes no mercado e muitas vezes já existentes nas empresas – como

sistemas de autenticação, armazenamento, redes ou *hypervisors* – facilitando assim a adoção e migração para sistemas de *Cloud*.

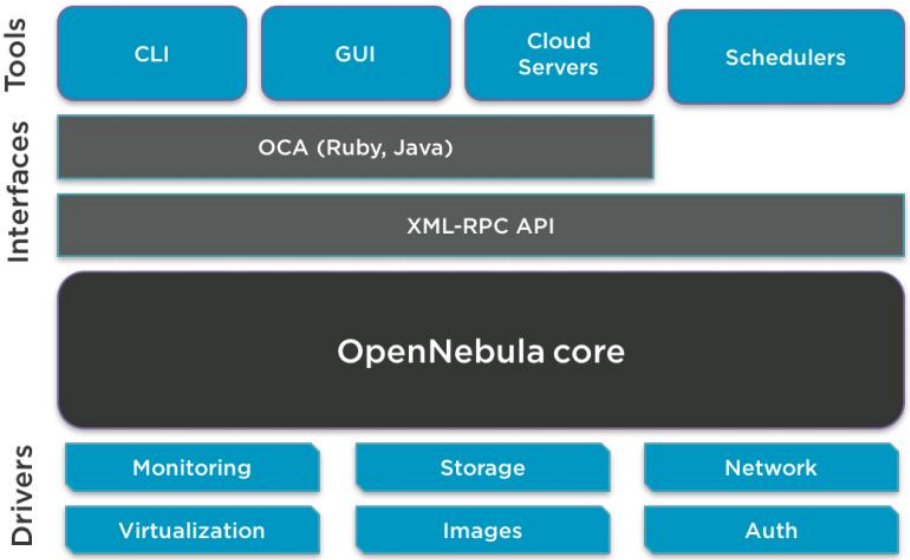

Figura 4.5 - Arquitetura conceptual *OpenNebula* (*OpenNebula Systems, 2016*)

Tools (Ferramentas): Camada exterior que disponibiliza ao utilizador as ferramentas, tais como linhas de comando ou interfaces gráficos através de páginas de Internet, para interagir com a infraestrutura de *Cloud*. É com recurso a estas ferramentas que o utilizador requisita os recursos pretendidos ou o administrador gere o ambiente e as máquinas virtuais. É nesta camada que são disponibilizados interfaces que permitem a comunicação com o exterior de modo a interligar com serviços de terceiros, por exemplo.

Interfaces / Core: Camada central com interfaces que promovem a integração com outros sistemas de mais alto nível que para tal dispõe de ligações para *Ruby*, *Java* ou *XMLRPC*. Esta camada gere o ciclo de vida completo das máquinas virtuais, desde a criação de uma rede virtual de forma dinâmica para atribuição de um endereço *IP* para uma *VM*, o que torna a solução de rede transparente para os utilizadores, à gestão do armazenamento de uma VM, como seja a atribuição de um disco para a mesma.

Drivers: É nesta última camada que se dá a comunicação com o sistema operativo da máquina física e onde é encapsulado o conjunto de recursos da infraestrutura de modo a torná-lo "transparente" para as camadas superiores. Neste nível estão serviços responsáveis pela criação, inicialização e eliminação de máquinas virtuais, alocação de recursos às mesmas e monitorização, tanto do sistema operativo da máquina física com das máquinas virtuais.

Ao contrário do que acontece com o *OpenStack* e o *Eucalyptus*, que possuem um sistema de ficheiros proprietário, como é o caso do *Swift* ou *Cinder* e o *Walrus*, nomeadamente, o *OpenNebula* utiliza o *NFS*, protocolo *standard* de utilização generalizada utilizado em variadíssimos outros sistemas que necessitam de aceder a ficheiros através de uma rede. O *OpenNebula* tem suporte para sistemas operativos *Linux* como o *CentOS*, o *RHEL* ou o *Ubuntu*, para o suporte a *hypervisors* tem uma oferta mais ampla que abrange o *KVM*, *Xen*, *VMware*, *Hyper-V*, *OpenVZ*, *VirtualBox* e ainda, não tanto como o *Eucalyptus* mas, com suporte para o *Amazon EC2*. Em termos de linguagens de programação tem suporte oficial para *Java*, *C++*, *Ruby* e *Python*.

4.5. OpenStack

O *OpenStack* teve a sua origem em 2010 num projeto conjunto entre a *NASA* e a *Rackspace* com o intuito de construir uma plataforma de *Cloud* que fosse *open source* (OpenStack, 2014). A plataforma de *Cloud*, atualmente gerida pela fundação com o próprio nome, evoluiu consideravelmente na oferta que recai sobre soluções de *IaaS* em que conta com a participação de mais de 600 empresas e mais de 62000 pessoas, tornando assim na maior comunidade que suporta uma solução de *Cloud open source*. A *OpenStack Foundation* tem como diretores elementos de empresas como por exemplo a *SUSE*, *HPE*, *Red Hat*, *IBM*, *Intel*, *Canonical*, *Rackspace*, entre outras (Companies, 2016).

Ao contrário do que acontece com outros fornecedores de serviços de *Cloud* como o *Eucalyptus* ou *OpenNebula*, em que disponibilizam a sua solução de forma unificada, o *OpenStack* é conhecido como sendo uma solução modular que, mediante a conjugação que é possível fazer com os seus vários serviços, coloca à disposição um leque variado de arquiteturas capazes de responder a diferentes necessidades de computação. A Figura 4.6 demonstra, numa perspetiva de alto nível, a arquitetura da plataforma.

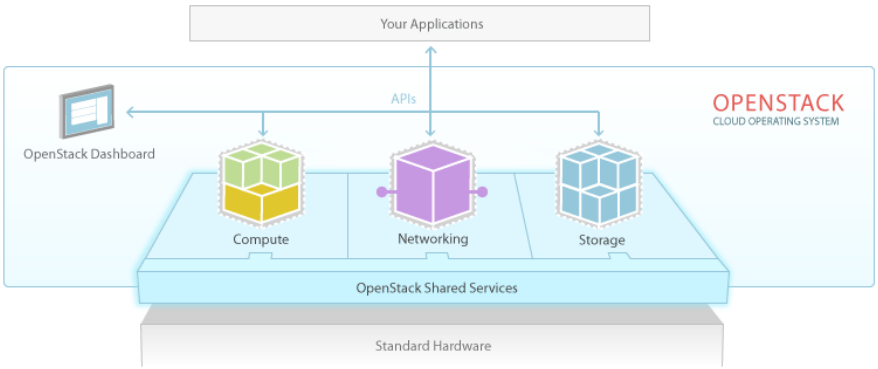

Figura 4.6 - Arquitetura dos principais serviços do *OpenStack* (*OpenStack, 2014*)

Os principais módulos de *software* que constituem uma infraestrutura de Cloud são:

Computação: De nome *Nova*, é o elemento principal da infraestrutura de *Cloud* em que recaem as funções de gestão da mesma, seja da atribuição de endereçamento *IP* e recursos como memória, processador ou disco a fornecer às *VM's*. A gestão das imagens com as quais são criadas as instâncias está a cargo do *Glance*. O suporte em termos de *hypervisors* passa pelo *KVM*, *Xen*, *Hyper-V*, *VMware* ou *LXC*.

Rede: Apelidado de *Neutron*, este módulo fornece elevadas capacidades de configurações e gestão de rede dentro da plataforma como balanceamento de carga, criação de novas topologias graficamente através do ambiente gráfico de gestão, criação de *VLAN*, roteamentos ou políticas de acesso.

Armazenamento: *Swift* é o componente do *OpenStack* responsável pelo armazenamento disponível na infraestrutura e gestão do mesmo, por seu lado o *Cinder* é responsável por fornecer às máquinas virtuais o equivalente a um disco para as máquinas físicas e que, em conjunto com o *Swift* pode ser utilizado para fazer cópias de segurança desses mesmos volumes.

Serviços Transversais: *Horizon* é o interface gráfico para gestão da infraestrutura e o *Keystone* é o responsável por implementar a segurança e autenticação transversal a todos os utilizadores e serviços internos.

Adicionalmente existem outros módulos de relevo tais como o *Trove* para bases de dados, o *Manila* para um sistema de ficheiros distribuído ou o *Ceilometer* para fornecer dados de telemetria. De todas as plataformas que apresentam soluções de *IaaS* é a que mais sistemas operativos suporta ativamente na comunidade, sendo este o *Ubuntu*, *RHEL*, *CentOS*, *Fedora*, *OpenSuse* ou o *OpenSuse Linux Enterprise Server*. Não tendo sido uma prioridade no seu desenvolvimento inicial, existem projetos a decorrer de modo a criar compatibilidade com o *EC2* e *S3* da *Amazon*.

4.6. Comparação entre fornecedores

Num mercado cada vez mais orientado para as tecnologias, e por consequência cada vez maior, o leque de oferta de plataformas de Cloud *open source* é diversificado e com um bom nível de maturidade. Todas as soluções apresentadas, e muitas outras existentes, operam no mesmo mercado, contudo, contrariamente aquilo que esta premissa possa transmitir, a competição não literalmente direta nem tão pouco sairá uma única vencedora visto que, para o mesmo mercado da *Cloud* cada uma das plataformas apresenta características que as diferenciam (*Brockmeier, 2012*).

A Figura 4.7 apresentada por (*Llorente, 2013*) representa de certa forma o que foi enunciado anteriormente. De um modo simples mostra como as plataformas de *Cloud* se posicionam no mercado levando em consideração as variáveis mais comuns, o nível de flexibilidade e o modelo, mais virado para a disponibilização de infraestrutura ou para uma melhor gestão dos recursos virtualizados de um *datacenter*.

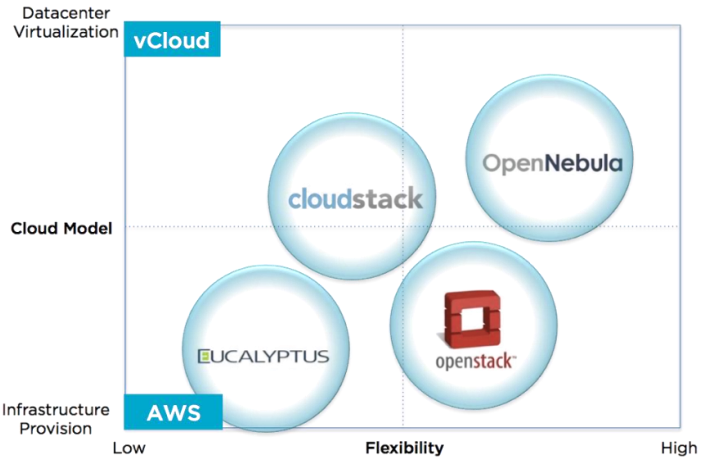

Figura 4.7 - Cenários em que se podem enquadrar as plataformas de *Cloud* (*Llorente, 2013*)

A principal característica da *Eucaliptus*, e um dos objetivos segundo os próprios, é disponibilizar uma plataforma o mais compatível possível com as soluções da *Amazon*. Faz desta uma plataforma menos flexível e mais focada para a infraestrutura do que por exemplo a solução apresentada pela *OpenNebula*.

O *OpenNebula*, que tem como um dos objetivos a criação de soluções de *Cloud* privadas, tem uma abordagem mais centrada para soluções de grandes *datacenters* onde tira proveito e

assenta em ambientes de armazenamento e soluções de rede de comunicação já estabelecidos. Esta abordagem trás vantagens do ponto de vista de quem desenvolve complementos ou módulos pois fica mais liberto das restrições e especificidades das plataformas de *Cloud* como acontece, por exemplos, em casos como o *OpenStack* ou da *Eucalyptus*.

Por outro lado a filosofia do *OpenStack* passa pela disponibilização da sua solução sob a forma de vários pacotes ou módulos de *software*, em que cada um gere um componente, e que devidamente configurados formam a infraestrutura de *Cloud* bastante flexível.

A *CloudStack* enquanto solução de *Cloud* encontra-se mais virada para as empresas e fornecedores de serviços que assentem a oferta na sua infraestrutura. Esta solução apresenta uma composição bastantes mais simples face ao *OpenStack*, por exemplo, e em termos de gestão de recursos virtuais encontra-se mais junto da *OpenNebula*.

Por fim temos a solução da *Nimbus*, não pior que todas as apresentadas, no entanto, pode-se dizer que seja um caso especial face à oferta existente visto centrar-se na exploração e investigação científica dotando a comunidade de infraestruturas hibridas que sejam capazes de responder às suas necessidades.

A Tabela 4.1 resume as características principais de cada uma das plataformas de Cloud analisadas.

Tabela 4.1 - Principais características em resumo das soluções de *Cloud*

	CloudStack	**Eucalyptus**	**Nimbus**	**OpenNebula**	**OpenStack**
Filosofia	Infra-estruturas públicas e privadas	Integração com Amazon	Investigação científica. Cloud comunitária	Infra-estruturas públicas e privadas	Infra-estruturas públicas e privadas
Sistemas Operativos	RHEL, CentOS e Ubuntu	RHEL 7 e CentOS 7	Linux / Unix (não é especificada nenhuma distribuição)	RHEL, CentOS e Ubuntu KVM, Xen,	RHEL, CentOS, Ubuntu, Fedora, OpenSuse e OSLES
Hypervisors	Xen, KVM, VMWare e Hyper-V	Xen, KVM, e VMWare	Xen e KVM	Xen, KVM, VMware, Hyper-V, OpenVZ e VirtualBox	Xen, KVM, VMware, Hyper-V e LXC
Linguagens programação	Java, Python, PHP e Perl	Java e PHP	Java e Python	Java, Python, C++ e Ruby	Java, Python, PHP, Ruby, C#
Armazenamento	NFS mas também compatível com OpenStack Swift	Walrus	Cumulus	NFS, SCP	Swift, Cinder
Licença	Apache License v2	GPL v3, Proprietário	Apache License v2	Apache License v2	Apache License v2

Todas as plataformas anteriormente enunciadas apresentam características interessantes e importantes para a implementação de uma infraestrutura de *Cloud* que se pretende robusta e flexível, capaz de se adaptar a ambientes complexos e ambientes simples. Das soluções analisadas decidiu-se optar pelo *OpenStack*. O facto de ser desenvolvido em *Python*, solução com maturidade e por ter uma gigantesca comunidade, ter suporte a inúmeros *back-ends* de armazenamento e de rede de diversos fabricantes (que possibilita a utilização do variado equipamento que empresa já possui) e possuir uma solução integrada de armazenamento baseado em objetos, foram fatores decisivos na sua escolha. O facto de existir um módulo específico para cada função, como é o caso da rede, outro módulo que gere as imagens, ou o armazenamento, faz com que o pré-requisito necessário seja somente ter o *hardware* e o sistema operativo instalado.

Aliado a isto existe o facto de alguns dos módulos ou componentes serem impulsionados por empresas de relevo na sua área, como é o caso do componente de rede que teve a colaboração da *Cisco*, juntando o facto de que há no mercado fornecedores de serviços como a *Rackspace*, a *UOL Host* ou a *HP*, que disponibilizam soluções de *Cloud* para empresas, revela uma certa confiança e aceitação desta ferramenta.

Neste capítulo foram apresentadas algumas soluções com as quais se podem implementar plataformas de *Cloud*. Foi dada uma breve explicação e enquadramento de cada uma delas tendo terminado com uma síntese das mesmas e a escolha do *OpenStack* como o pilar para a implantação da infraestrutura de *Cloud* privada que serve de base a este estudo. No capítulo que se segue irá ser apresentado em detalhe a plataforma do *OpenStack*, desde a sua arquitetura, os módulos de *software* que a constituem e o modo como estes se interligam.

5. OPENSTACK

No capítulo anterior foram apresentadas algumas plataformas sobre as quais seria possível implementar uma infraestrutura de Cloud, em que no final foi tomada a decisão, com base no facto de ser desenvolvido em *Python*, solução com maturidade e por ter uma gigantesca comunidade, ter suporte a inúmeros *back-ends* de armazenamento e de rede de diversos fabricantes (que possibilita a utilização do variado equipamento que empresa já possui) e possuir uma solução integrada de armazenamento baseado em objetos e no que se pretendia para este projeto, que era encontrar uma plataforma que fosse modular, passível de se adaptar a *hardware* de gama doméstica e ainda assim apresentar um desempenho capaz de virtualmente substituir um equivalente físico. Foram estes os fatores decisivos que pesaram na decisão pelo OpenStack.

Este capítulo pretende abordar um dos objetivos do trabalho, de entre as inúmeras ofertas de mercado e a dispersão de informação, dar a conhecer uma plataforma e dotar os leitores do conhecimento necessário para construir a sua própria infraestrutura de *Cloud*. Esta informação é colmatada e concretizada no capítulo seguinte, onde é apresentada a implementação feita no âmbito deste trabalho.

À data da implementação prática deste projeto a versão considerada estável pelo *OpenStack* tinha o nome de *Havana*. O lançamento de novas versões é feito em ciclos de seis meses que, após o lançamento inicial, passa por outros estados até ser considerada estável. Em cada lançamento são corrigidos problemas encontrados anteriormente e consoante o nível de desenvolvimento, lançados novos módulos, ou como são chamados na documentação do *OpenStack*, serviços. A versão *Havana* conta com os seguintes módulos: *Nova*, *Glance*, *Swift*, *Horizon*, *Keystone*, *Neutron*, *Cinder*, *Heat* e *Ceilometer*, todos estes, nomes de código pelos quais são conhecidos dentro do projeto OpenStack. A solução reconhece duas entidades lógicas, o "*Cloud Controler*" como sendo o elemento principal que gere toda a infraestrutura e o "*Compute Node*", servidor ou conjunto de servidores que disponibilizam toda a capacidade de computação existente na IaaS.

A Tabela 5.1 apresenta o nome das versões, datas que foram lançadas e os serviços que foram sendo disponibilizados em cada uma delas.

Tabela 5.1 - Histórico de versões do *OpenStack* (*OpenStack Releases, 2016*)

Nome	Lançamento	Componentes
Austin	21-10-2010	Nova, Swift
Bexar	03-02-2011	Nova, Glance, Swift
Cactus	15-04-2011	Nova, Glance, Swift
Diablo	22-09-2011	Nova, Glance, Swift
Essex	05-04-2012	Nova, Glance, Swift, Horizon, Keystone
Folsom	27-09-2012	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Grizzly	04-04-2013	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Havana	17-10-2013	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer

Nome	Lançamento	Componentes
Icehouse	17-04-2014	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove
Juno	16-10-2014	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara
Kilo	30-04-2015	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara, Ironic
Liberty	15-10-2015	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara, Ironic, Zaqar, Manila, Designate, Barbican, Searchlight
Mitaka	07-04-2016	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara, Ironic, Zaqar, Manila, Designate, Barbican, Searchlight, Magnum
Newton	06-10-2016	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara, Ironic, Zaqar, Manila, Designate, Barbican, Searchlight, Magnum, aodh, cloudkitty, congress, freezer, mistral, monasca-api, monasca-log-api, murano, panko, senlin, solum, tacker, vitrage, watcher

5.1. Arquitetura Conceptual

O OpenStack consiste num agregado de módulos independentes, chamados de serviços, que necessitam de se validar e autenticar centralmente para que possam comunicar entre si. A menos que a ação a realizar necessite de privilégios de administrador, esta comunicação é feita através de *API's* públicas. A figura que se segue representa a arquitetura conceptual do *OpenStack Havana* onde está descrito a relação existente entre cada um destes serviços ou módulos.

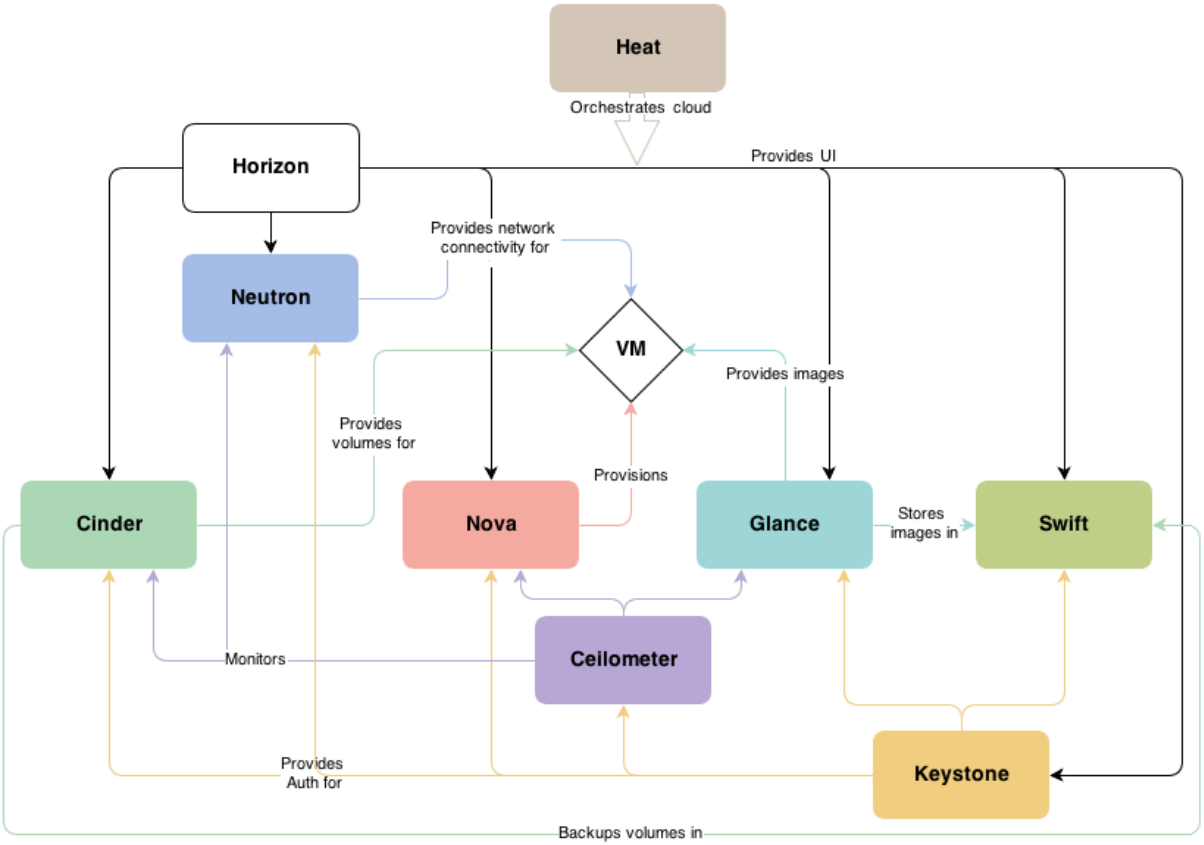

Figura 5.1 - Arquitetura Conceptual do *OpenStack Havana* (*OpenStack, 2014*)

A sua arquitetura modular e altamente configurável permite que se desenhe uma solução de acordo com os recursos de *hardware* disponíveis, sejam eles profissionais ou de gama comercial, haja ao dispor muitos ou poucos equipamentos.

Os serviços do *OpenStack* estão organizados, conforme é possível ver na Figura 4.6 que acompanha a descrição da plataforma no capítulo anterior, segundo grupos consoante o papel que desempenham na implementação da solução. Estes grupos e consequentes componentes que os constituem são explicados de seguida.

5.1.1.Compute

A computação é o conjunto de software principal da solução, ditando a organização e centralizando as interações dos serviços, que serve de base para a infraestrutura de *Cloud*. Tem no serviço *Nova* o elemento chave e conta com os serviços do *Glance* para as imagens.

- **Nova** (*Compute Layer*)

Também conhecido como *OpenStack Compute* ou *Nova Compute*, este serviço é o coração da arquitetura de Cloud, essencial para uma implementação básica, através do qual é feita a gestão do ciclo de vida das instâncias. O agendamento da criação, execução e términus de máquinas virtuais num dos nós disponíveis na infraestrutura é feito através de drivers que comunicam com a camada de virtualização onde se encontram *hypervisors* como o *KVM*, *Xen*, *Hyper-V*, *VMware* ou *LXC*. É uma matriz de *software* que fornece serviços para a gestão de recursos da nuvem através de *API's*, capaz de orquestrar instâncias em execução, redes e controlo de acessos, com facilidade para ser escalável horizontalmente.

- **Glance** (*Image Management*)

Este serviço atua como um registo, um catálogo de imagens existentes por forma a serem utilizadas na plataforma como base para novas instâncias. Os utilizadores podem adicionar novas imagens ou fazer cópias de instâncias existentes. Essas cópias ficam registadas e podem servir de base para a criação de novos servidores. As imagens catalogadas podem ser armazenadas no serviço *Swift*, bem como em outros locais, por exemplo, num sistema de ficheiros simples ou servidores *web* externos. De entre outros serviços que interagem com as imagens, é o único que pode adicionar, apagar, partilhar ou duplicar imagens. O suporte em termos de formatos das imagens é grande, podendo ser em *Raw*, *ISO*, *QCOW2* (*Qemu/KVM*), *VHD* (*Hyper-V* entre outros), *VDI* (*Qemu / VirtualBox*), *VMDK* (*VMWare*) ou formatos da *Amazon* (*AKI / AMI / ARI*).

5.1.2.Networking

A criação de um módulo específico para a configuração e gestão dos esquemas de rede da infraestrutura só aconteceu em 2012 com a versão *Folsom* do *OpenStack* que, até então, estavam a cargo do *Nova*. Embora continue a poder ser utilizado na configuração e gestão de rede, a criação de um novo projeto, de seu nome *Neutron*, veio trazer maior versatilidade e possibilitar configurações mais avançadas.

- **Neutron** (*Networking*)

Inicialmente criado sob o nome de *Quantum*, este só perdurou duas versões tendo sido renomeado para *Neutron* em 2013 com a versão *Havana* do *OpenStack*. A renomeação deste projeto deveu-se a um conflito com outra empresa devido a marcas registadas sob esse nome.

De modo resumido e muito simplista, é um sistema para a gestão de rede e endereços *IP*. Para os outros módulos internos do *OpenStack*, como é o caso do *Nova* que faz a gestão da computação e consequentemente a comunicação com outros componentes, o *Neutron* disponibiliza a rede como um serviço. Este modelo permite ao utilizador desenhar os seus esquemas e configurações de rede desde *IP's* estáticos, atribuídos por *DHCP*, configurações com recurso a *VLAN's*, grupos de rede ou ter servidores a ligarem com mais do que uma rede. A sua arquitetura foi desenhada de modo a suportar funcionalidades mais avançadas como sistemas de deteção de intrusão, balanceamento de carga e *VPN's*.

5.1.3. Storage

Quando se projeta uma infraestrutura de *Cloud* já com alguma dimensão, é natural que se equacione serviços e servidores dedicados para a gestão do espaço de armazenamento. Para tal o *OpenStack* disponibiliza o *Swift* e o *Cinder* que são explicados de seguida.

- **Swift** (*Object Storage*)

É um sistema de armazenamento altamente redundante e escalável. A sua arquitetura distribuída facilita o escalonamento horizontal, bastando para o efeito adicionar novos servidores ao *cluster*. No caso da redundância, esta é obtida pelo *software* do serviço, o qual é responsável pela replicação e integridade da informação de toda a estrutura. Este serviço pode ser utilizado pelo *Cinder* para fazer cópias de segurança dos volumes das máquinas virtuais.

- **Cinder** (*Block Storage*)

Também conhecido por *OpenStack Block Storage* ou Cinder *Volumes*, este serviço é responsável pela gestão de discos rígidos virtuais da infraestrutura. Pode trabalhar em conjunto com o *Swift* para guardar cópias de segurança dos volumes. Em versões anteriores do *OpenStack* este serviço era uma parte integrante do *Nova* sob a forma do componente *Nova-volume*. O *Cinder* irá interagir principalmente com o *Nova* ao fornecer volumes para as suas instâncias, permitindo através da sua *API* a manipulação de volumes, tipos de volume, anexação, eliminação dos mesmos e cópias de máquinas virtuais (*snapshots*).

5.1.4. Shared Services

Estes serviços, também eles essenciais para a construção de uma infraestrutura de Cloud, são transversais a toda a solução, pelas funções que desempenham e interação com os restantes serviços da solução.

- **Keystone** (*Identity*)

Serviço transversal à infraestrutura, este é responsável pela autenticação e autorização de todos os elementos da solução. Sejam eles utilizadores ou serviços, necessitam de estar registados e autorizados no *Keystone* de modo a que possam interagir e dessa forma prestar o

serviço de *Cloud* esperado. O *Keystone* para além de integrar com outros sistemas, como é o caso do *LDAP*, suporta várias formas de autenticação, incluindo credenciais compostas por nome de utilizador e palavra-chave ou sistemas baseados em *token*.

- **Horizon** (*Dashboard / User Interface*)

O *Horizon* disponibiliza o interface gráfico, acessível através de um *browser*, para utilizadores finais ou administradores do sistema. Este serviço concentra num único local, de forma simples e intuitiva, todas as funções disponibilizadas por outros serviços da infraestrutura, tais como, criação de *VM's*, gestão de acessos e permissões, gestão de discos virtuais ou atribuição de endereçamento *IP* às instâncias. Este painel é uma das várias formas pelas quais os utilizadores podem interagir com a solução, existindo ao dispor as linhas de comandos, ou *API's* públicas para cada um dos serviços o que permite o desenvolvimento de ferramentas personalizadas.

- **Ceilometer** (*Telemetry*)

É um novo componente que surgiu com o *OpenStack Havana* e que fornece serviços de monitorização da infraestrutura. Recolhe um vasto leque de dados com o propósito de criar relatórios para faturação, realizar ações com vista à escalabilidade, avaliação de desempenho da solução ou análise estatística.

- **Heat** (*Orchestration*)

Mais um componente novo do *OpenStack* que surgiu com a versão *Havana*. Este serviço permite orquestrar a infraestrutura de *Cloud*. Com base em *templates*, permite a automatização de ações que se prevê serem feitas com alguma frequência. É possível agilizar a execução de máquinas virtuais com um determinado conjunto de parâmetros como memória, disco, processadores ou imagem ou automatizar ações sobre a infraestrutura com base em dados recolhidos pela monitorização.

5.1.5.Supporting Services

Por serviços de suporte são entendidos aqueles que de algum modo suportam as funcionalidades prestadas pelos módulos/serviços da infraestrutura.

- **Base de Dados**

Por defeito o *OpenStack* utiliza o *MySQL* como o seu sistema de gestão de base de dados para que cada um dos seus serviços possa armazenar as suas configurações e informação importante para a gestão da plataforma. Por norma a base de dados, *MySQL*, *MariaDB* ou *PostgreSQL* é instalada no servidor principal da infraestrutura, o *Controller Node*, e servida uma base de dados para cada um dos serviços.

- **AMQP**

AMQP é o protocolo de mensagens usado no *Openstack* para comunicação entre processos. Ele usa por padrão o *RabbitMQ* e suporta outros como *Qpid* ou *ZeroMQ*. A ferramenta utilizada fornece uma plataforma comum para que os processos enviem e recebam mensagens entre eles,

fazendo uso de pouco ou nenhum conhecimento das definições dos componentes de cada um. É um serviço essencial para uma implementação básica de arquitetura de *Cloud*.

5.1.6. Novos projetos e funcionalidades

A arquitetura e serviços que foram explicados até ao momento referem-se à versão *Havana* do *OpenStack*, versão utilizada para a criação da infraestrutura de *Cloud* que serve de base a este trabalho. Desde então nasceram e foram colocados em produção novos projetos, outros ainda como protótipos e outros ainda em fase de integração com serviços já existentes, servindo este subtópico para apresentação dos mesmos, face ao que existe na documentação. A Figura 5.2 representa a arquitetura conceptual atualizada do *OpenStack Newton* onde estão representados os mais recentes módulos de *software* disponíveis para integrar numa solução de *IaaS*.

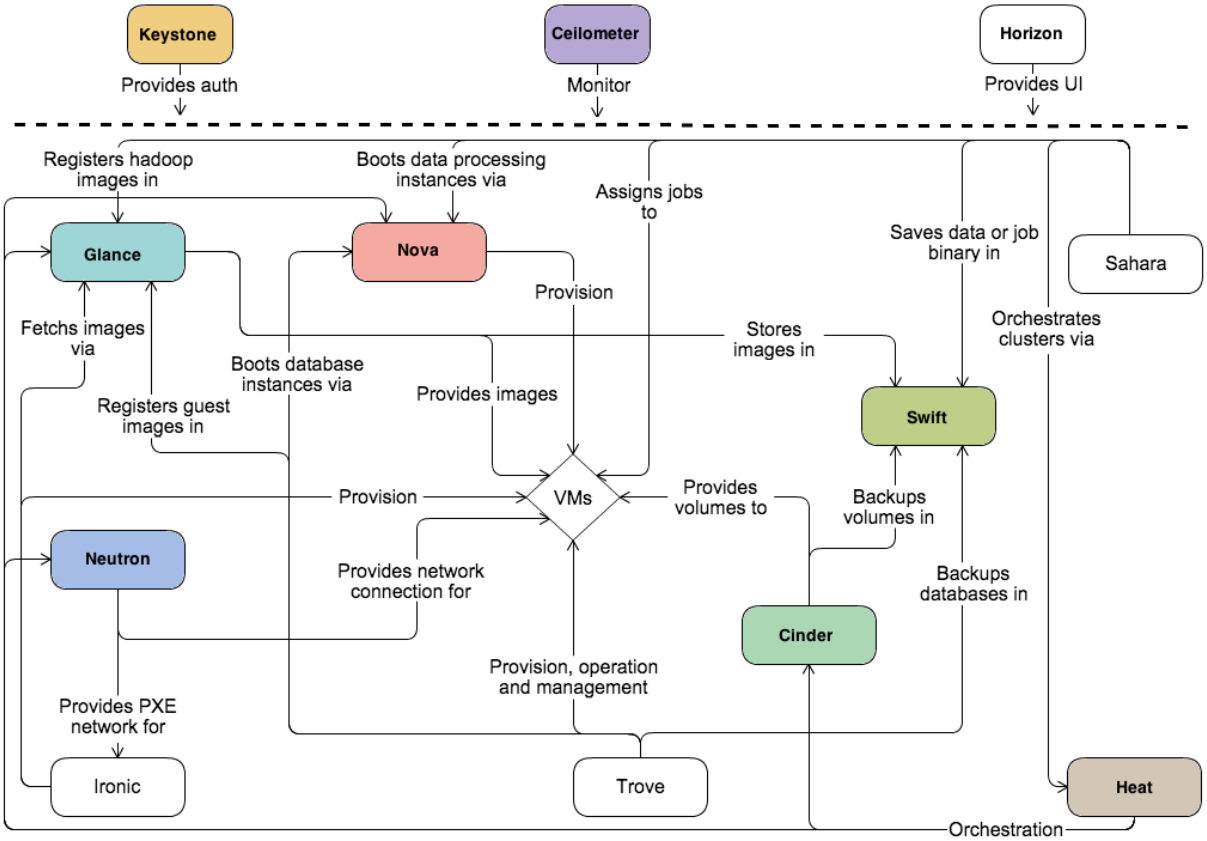

Figura 5.2 - Arquitetura Conceptual do *OpenStack Newton* (*OpenStack, 2014*)

Os serviços que apareceram após a implementação deste projeto encontram-se em diferentes fases de maturação e aceitação pelo que, é recomendável aceder à página do *OpenStack* por forma a confirmar o estado atual de cada um. De acordo com a documentação, para além do *Horizon*, *Ceilometer* e *Heat*, os serviços que se apresentam de seguida, são considerados como opcionais e não como componentes *core* da arquitetura.

- **Trove** (*Database*)

É o nome dado ao projeto que visa disponibilizar *dBaaS* com suporte para bases de dados relacionais e não relacionais.

- **Sahara** (*Elastic Map Reduce*)

Componente que fornece recursos para se construir um *cluster Hadoop* no *OpenStack*. Aos utilizadores é-lhes pedido que especifiquem parâmetros tais como a versão do *Hadoop*, a topologia a utilizar e detalhes da capacidade de computação que pretendem, desde memória, disco e processador.

- **Ironic** (*Bare-Metal Provisioning*)

Ironic é um projeto do *OpenStack* que fornece máquinas físicas em vez de máquinas virtuais através da *Cloud*. Há algumas situações em que é mais vantajoso face à virtualização, como por exemplo, funcionalidades que necessitem de acesso a *hardware* que não pode ser virtualizado ou bases de dados que tem melhor desempenho em máquinas não geridas por um *hypervisor*.

- **Zaqar** (*Messaging Service*)

É um serviço de mensagens e notificações criado com vista aos programadores de sistemas *web* e móvel. Este serviço é muito semelhante aos já existentes da *Amazon SQS* (*Simple Queue Service*) e *SNS* (*Simple Notification Service*). Ao contrário do sistema de filas de mensagens implementado por exemplo pelo *RabbitMQ* em que utiliza o protocolo *AMQP* o *Zaqar* é uma *API* que pretende ser de mais fácil manuseio e compreensão face às *API's* sobre as quais assenta para comunicar com os *back-ends* dos serviços

- **Manila** (*Shared Filesystems*)

Este projeto deriva em parte do já conhecido *Cinder*. É um projeto independente que fornece um sistema de ficheiros partilhado e distribuído. Embora a sua utilização primária seria fornecer armazenamento de ficheiros acessível entre as instâncias do *OpenStack Compute*, pretende-se que este seja um projeto que forneça um serviço de forma independente de modo a aceitar outros sistemas de ficheiros.

- **Designate** (*DNS Service*)

É uma *API* baseada em *ReST* que fornece *DNS* como um serviço para a plataforma do *OpenStack*, sendo compatível com outros *back-ends*, tais como *PowerDNS* e *BIND*. A sua finalidade passa por interagir com servidores *DNS* dotando o administrador de um método simplificado de criar *DNS* para uma ou mais zonas dentro da infraestrutura.

- **Barbican** (*Key Management*)

O *Barbican* é uma API desenhada para o armazenamento seguro, disponibilização e gestão de informação sensível e crítica tal como palavras-chave, chaves de encriptação e certificados *X.509*.

- **Magnum** (*Containers*)

É um serviço desenvolvido pela equipa de *Containers* do *OpenStack* com o intuito de disponibilizar aos utilizadores mecanismos de orquestração de *Containers* como é o caso do *Docker* ou *Kubernetes*. Ele depende do serviço interno do *OpenStack* para a orquestração, o *Heat*, tirando assim um maior proveito das ferramentas referidas.

- **Murano** (Application Catalog)

O projeto *Murano* introduz um catálogo de aplicações para o *OpenStack*, permitindo que programadores e administradores publiquem vários aplicativos prontos para a *Cloud* num catálogo categorizado navegável. Os utilizadores da infraestrutura, incluindo os menos experientes, podem utilizar esse catálogo para selecionar a aplicação e assim criar os seus próprios ambientes, por exemplo servidores *web*.

- **Congress** (*Governance*)

Congress é um projeto através do qual é possível implementar e gerir políticas para a gestão de aplicações e infraestrutura de *Cloud*. Garantir por exemplo que uma determinada instância é adicionada a uma determinada rede ou que uma aplicação quando executada está protegida por uma *firewall*.

5.2. Arquitetura Lógica

Sendo agora mais específicos na abordagem à arquitetura da infraestrutura, o que implica aumentarmos o nível de detalhe e por consequência a complexidade do diagrama, ficamos a conhecer quais os serviços internos que compõem cada um dos módulos. Saber bem a que se propõe a infraestrutura e conhecer os serviços disponíveis e as suas funções facilita o processo de planear e escolher os elementos necessários, visto que nem todos são obrigatórios para a construção de uma solução de *Cloud*.

A Figura 5.3 ilustra a arquitetura mais comum utilizada para criar uma *IaaS* baseada no *OpenStack* – assumindo que o administrador do sistema irá utilizar todos os serviços em conjunto na sua configuração mais usual – no entanto, devido à sua modularidade e variedade de tecnologias que têm vindo a ser implementadas, esta não é a única configuração possível. É sim uma configuração que faz uso dos projetos de *software OpenStack* mais maduros e estáveis, o que proporciona uma boa base de conhecimento e as ferramentas necessárias para que o leitor consiga explorar e desenhar a sua própria solução.

5.2.1. Nova (Compute Layer)

Abaixo estão descritos os serviços internos do componente de computação Nova e uma breve descrição das suas funções.

`nova-api`: Recebe os pedidos de máquinas virtuais, encaminha-os e dá resposta aos utilizadores. Suporta a *API* da *Amazon EC2* e é responsável pelo tratamento dos pedidos de orquestração da *Cloud*.

`nova-api-metadata`: Aceita pedido de meta dados por parte das instâncias. Este serviço é usualmente instalado quando a rede é gerida pelo *Nova* e em configurações *multi-host* (em que cada nó tem o serviço *nova-network* instalado, de modo a permitir o funcionamento dos nós em caso de falha do *Cloud Controller*).

`nova-compute`: Serviço que gere o ciclo de vida das *VM's*, criação e eliminação destas através dos drivers de comunicação com o *hypervisor*, como é o caso do *libvirt* para o *KVM* ou *QEMU* ou o *XenAPI* para o *XenServer*.

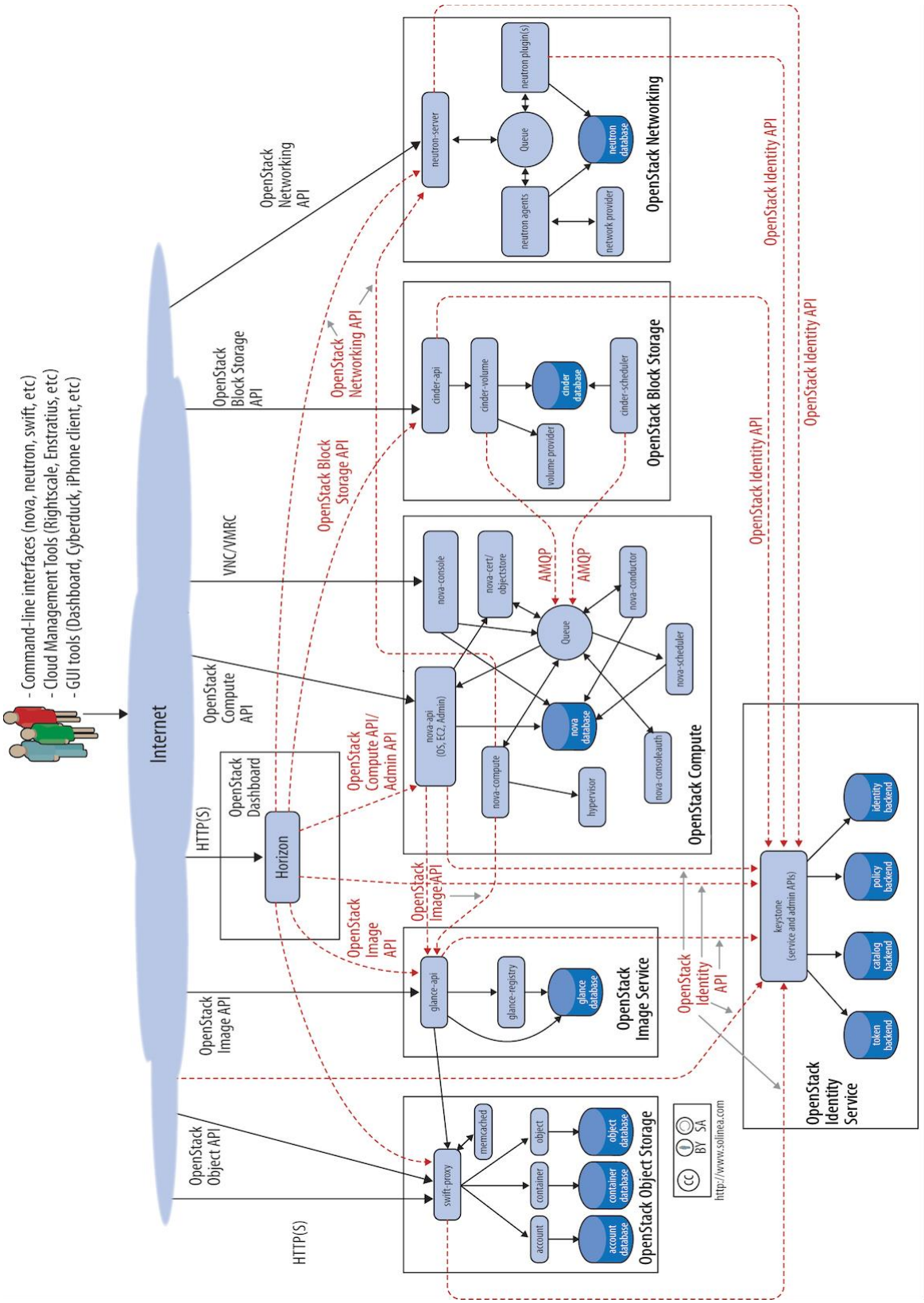

Figura 5.3 - Arquitetura Lógica do *OpenStack Havana* (*Fifield et al., 2014*)

`nova-scheduler`: Serviço de gestão de tarefas. Apanha um pedido da fila de espera e atribui a execução da máquina virtual a um determinado servidor.

`nova-conductor`: É um módulo criado para mediar as interações do *nova-compute* com a base de dados.

`nova-console`, `nova-consoleauth`, `nova-novncproxy`, `nova-xvpnvncproxy`: Serviços que estão relacionados com a necessidade de se estabelecer um *proxy* e de validar *tokens* para que os utilizadores possam aceder às suas instâncias, seja por consola ou VNC.

`nova-network`: Aceita pedidos fila de mensagens e executa tarefas a fim de responder a esses pedidos, tais como manipulação de rotas, atribuição de *IP's* ou configuração de portas.

`nova-cert`: Serviço que permite a plataforma gerar e validar certificados *x509* utilizado entre outros para comunicar com a *API* da *Amazon EC2*.

`nova-objectstore`: Usualmente utilizado em configurações que necessitem de utilizar as ferramentas euca2tools (ferramentas de linha de comando do *Eucalyptus*) para compatibilidade com o *Amazon S3*. O serviço do *Nova* interpreta os pedidos vindos do *euca2tools* e traduz para pedidos internos do *OpenStack*.

5.2.2.Glance (*Image Management*)

Em seguida são descritos os serviços internos do componente responsável pela gestão das imagens e uma breve descrição das suas funções. Conjuntamente com o *Nova*, o *Glance* ocupa uma posição central e de relevo para o funcionamento da *IaaS*.

`glance-api`: Serviço principal do projeto de *software* que recebe os pedidos de referente às imagens com as quais serão criadas as máquinas virtuais.

`glance-registry`: Serviço responsável por armazenar e tratar meta dados referentes às imagens, como o seu tamanho ou tipo.

`Repositório para ficheiros das imagens`: O *Glance* suporta vários tipos de repositórios, como o próprio sistema de ficheiros do servidor onde está instalado o serviço, o próprio *Swift* ou o *Amazon S3*, entre outros.

A par com o *Nova*, o *Glance* representa um papel de relevo para a solução de *IaaS* do *OpenStack*. As suas principais interações são com o *Nova* e a responder a pedidos de informação e gestão das imagens através da sua *API*.

5.2.3.Neutron (*Networking*)

Vamos agora descrever os serviços internos do componente de rede *Neutron* acompanhado de uma breve explicação das suas funções:

`neutron-server`: Aceita pedidos que são feitos através da sua API e encaminha-os para o *plugin* apropriado para o executar.

`Plugins de rede e agentes`: Ligam e desligam portas, manipulam endereçamento de rede ou disponibilizam *IP's*. O *OpenStack* de base tem *plugins Layer3*, *DHCP* e agentes

específicos para *switches* físicos e virtuais da *Cisco* (principal contribuidor deste projeto), *NEC OpenFlow*, *Open vSwitch*, *Linux bridging*, e o *VMware NSX*.

A principal interação deste serviço é feita com o *Nova*, para onde fornece rede e conectividade para as suas instâncias.

5.2.4.Swift (*Object Storage*)

O *Swift* é um projeto de *software* bastante completo e altamente distribuído com uma arquitetura mínima que implica cinco servidores, em que um é utilizado para a gestão do ambiente propriamente dito e os restantes para replicar a informação e assim manter a consistência em caso de falha. Encontram-se descritos de seguida os serviços internos do *Swift*:

`swift-proxy-server`: Aceita pedidos para carregamento de ficheiros, modificar meta dados ou criação de pastas ou *containers*. Fornece ficheiros e listagens de pastas para páginas *web*.

`swift-account-server`: Gere as contas criadas com o gestor de ficheiros.

`swift-container-server`: Faz a gestão de pastas ou *containers* do sistema de armazenamento.

`swift-object-server`: Faz a gestão dos ficheiros da infraestrutura de armazenamento.

`swift client`: Permite aos utilizadores, sejam eles administradores ou utilizadores criados no *Swift*, enviarem pedidos para a plataforma de armazenamento através de linhas de comandos.

`swift-init`: Utilizado para realizar tarefas de gestão sobre outros componentes do *Swift*, como reiniciar ou parar serviços.

`swift-recon`: Ferramenta que recolhe e disponibiliza informações de monitorização da solução de armazenamento.

`swift-ring-builder`: Áreas lógicas criadas dentro do cluster que irá permitir ao *Swift* gerir a informação em grupos no lugar de ficheiros individualmente.

É comum ser utilizado para armazenar os ficheiros das imagens da infraestrutura, comunicando assim com o *Glance*, ou servir páginas estáticas e ficheiros através de *HTTP*.

5.2.5.Cinder (*Block Storage*)

O sistema de gestão de blocos, ou discos virtuais que se ligam às máquinas virtuais, é descritos de seguida, onde se apresentam os seus serviços internos:

`cinder-api`: Recebe os pedidos através da sua *API* e encaminha-os para execução pelo *cinder-volume*.

`Cinder-volume`: Executa os pedidos de discos virtuais e atualiza a informação dos mesmos na base de dados. O *Openstack* tem drivers que permitem ao serviço interagir com outros fornecedores de armazenamento como a *IBM*, *SolidFire* ou *NetApp*, entre outros.

`cinder-scheduler`: À semelhança do seu homologo no *Nova*, o *scheduler* irá escolher, caso existam vários, qual o melhor repositório onde irá criar o volume para posteriormente associar a uma máquina virtual.

`cinder-backup daemon`: Processo que permite à plataforma criar cópias de segurança dos volumes.

Tal como o serviço de rede, o *Cinder* interage maioritariamente com o *Nova* a fim de disponibilizar discos virtuais para as suas instâncias.

5.2.6. Keystone (*Identity*)

O projeto Keystone é um serviço centralizado que gere o sistema de autenticação e autorização da plataforma para os serviços internos e utilizadores, podendo ainda integrar com outros sistemas de autenticação como o LDAP. Toda a integração com os restantes serviços e com a plataforma é feita com a instalação do pacote Linux no servidor principal da plataforma (Controller Node)e identificados os pontos de acesso (API's) para a interação com os restantes serviços e configurada a base de dados do projeto, que permite assim saber quais os serviços existentes e dar-se a conhecer como sendo o sistema onde se validam para comunicar.

5.2.7. Horizon (Dashboard / User Interface)

O pacote de instalação deste projeto disponibiliza os ficheiros que constituem o interface *web*. É uma aplicação baseada em *Django* e que assenta no servidor *web Apache*.

Após configuração da base de dados e do servidor *web* com informação específica da infraestrutura criada, tal como endereçamentos *IP* ou nome do servidor principal e credenciais geradas no serviço de autenticação, a gestão da solução de *IaaS* através de um ambiente gráfico fica acessível a partir de uma página *web*.

5.3. Arquitetura de Serviços

Até ao momento foram apresentados os vários serviços do *OpenStack*, desde os fundamentais para alicerçar a infraestrutura de Cloud até aqueles que podem ser considerados opcionais para o funcionamento da mesma. No final, a infraestrutura de *Cloud* privada que será construída irá disponibilizar poder de computação ao utilizador final sob a forma de máquinas virtuais. Dentro de cada um dos serviços do *OpenStack*, também chamados de módulos ou projetos de *software*, existem vários serviços individuais que em conjunto com outros formam uma cadeia de ações das quais resulta a disponibilização das máquinas virtuais. Pretende-se com este subtópico dar a conhecer, numa perspetiva de alto nível, qual o processo que é desencadeado após um pedido de recursos por parte do utilizador.

O *Nova Compute* é o componente de *software* principal da solução *OpenStack* que determina a organização e as interações dos serviços.

Os principais intervenientes nas ações tomadas sobre a *IaaS* são nomeadamente a *API*, a fila de mensagens (*Queue*), os servidores de computação (*Compute Worker*), o serviço que controla a comunicação de rede (*Network Controller*), o sistema de gestão de volumes (*Volume Controller*), o serviço responsável pela gestão das tarefas (*Scheduler*) e pela gestão das imagens (*Image Store*).

A *API* disponibiliza aos utilizadores a gestão e controlo das configurações do *hypervisor*, do armazenamento e da rede através do painel de controlo facilmente acedido através do seu interface gráfico *web*. A fila de mensagens é o mediador da comunicação entre serviços internos como é o caso, por exemplo, do *Compute Node*, *Volumes* ou *Scheduler* entre outros.

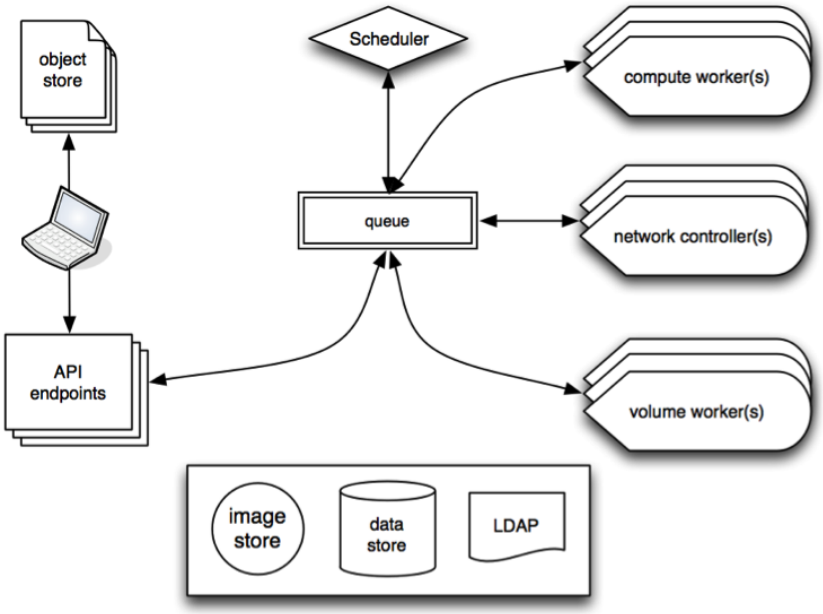

Figura 5.4 - Arquitetura dos principais serviços (*OpenStack Havana, 2014*)

De modo genérico, uma troca típica de mensagens começa pela autenticação do utilizador na *API* por forma a garantir que este tem autorização para operar sobre a infraestrutura. Seguidamente é feita uma avaliação do pedido quanto à disponibilidade dos recursos solicitados e, em caso afirmativo, é dado o devido seguimento para execução do mesmo. Posteriormente é devolvida uma mensagem à origem para que a *API* informe o utilizador face ao que este solicitou à infraestrutura. Estes pedidos podem ser de vários tipos, desde atribuição de *IP* público a um servidor, criação de volumes, ou cópia de uma instância, não exclusivamente a execução de uma nova instância ou servidor.

A Figura 5.5 representa o caso concreto do pedido de criação de uma máquina virtual no *OpenStack*, onde podem ser vistos os passos e interações do processo.

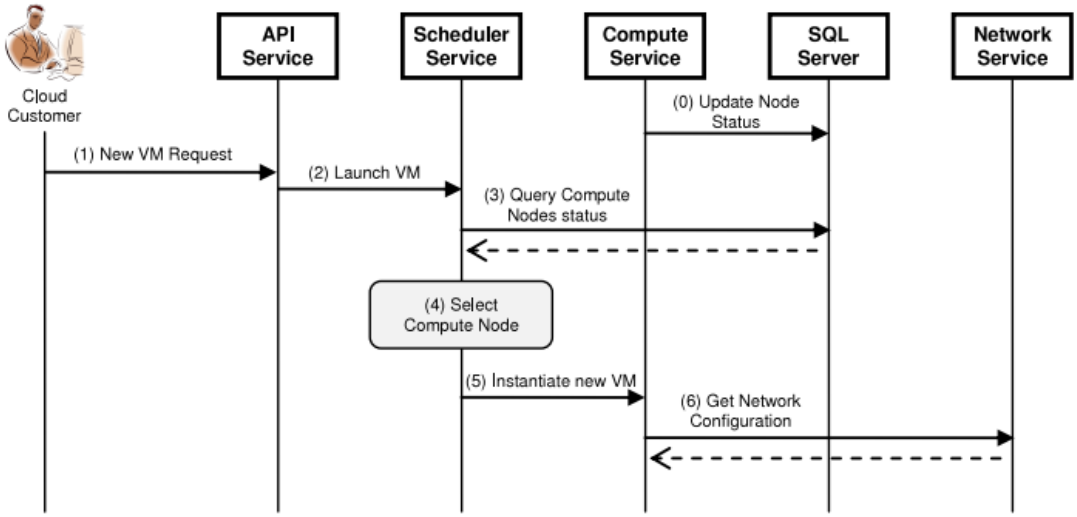

Figura 5.5 - Pedido de criação de uma máquina virtual no *OpenStack* (*Corradi, 2012*)

O passo inicial (passo 0) é realizado periodicamente por vários serviços da infraestrutura, sendo que, neste caso, a base de dados é atualizada com informação de máquinas virtuais ativas e capacidade dos servidores físicos. Quando é feito o pedido de uma nova instância (passo 1), a *API* envia-o para o gestor de tarefas da infraestrutura, o *Scheduler* (passo 2). A informação recolhida sobre os servidores disponíveis é importante nesta fase (passo 3) pois serve de base para a decisão de qual o servidor físico se deve atribuir (passo 4). Após ser atribuído um servidor é enviada ordem de execução da *VM* para o serviço de computação do respetivo nó (passo 5) que por sua vez solicita ao serviço de rede as configurações da rede interna de serviço, utilizada para a comunicação entre *VM´s* e a plataforma (passo 6).

5.4. Modelo de Objetos

Ao olhar para a solução de *Cloud* na ótica do utilizador final, ao contrário do que tem sido feito até ao momento, em que tem sido colocado o foco no administrador da infraestrutura, conseguem-se identificar várias entidades ou objetos que fazem parte do ambiente e com os quais se tem de interagir. Esses objetos estão representados na Figura 5.6 que simultaneamente mostra como se posicionam perante a solução como um todo.

Temos os utilizadores que acedem e controlam a solução de *Cloud* e as respetivas chaves de autenticação a fim de tornar o ambiente seguro e confiável. Os utilizadores podem estar associados a mais do que um projeto, como é de esperar que cada projeto tenha vários utilizadores. Projetos são áreas privadas e podem ser vistos como pequenas *Clouds* dentro da *Cloud* privada da empresa, um espaço único adaptado à medida das necessidades de diferentes departamentos, dando-lhes privacidade e liberdade sem prejudicar o trabalho uns dos outros. Cada um possui um conjunto de *IP's* públicos para que as *VM's* possam aceder ao exterior, um conjunto de regras de segurança, instâncias (máquinas virtuais) e volumes de armazenamento que por sua vez podem posteriormente ser associados a *VM's*.

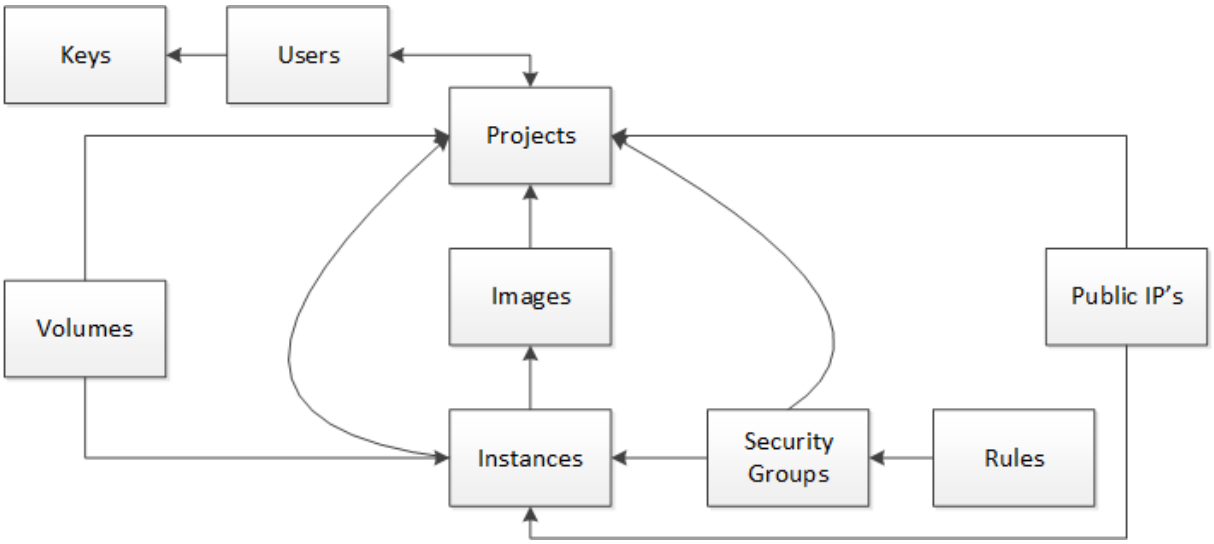

Figura 5.6 - Modelo de objetos do *OpenStack* (*Get started, 2015*)

São estes os elementos que, na ótica do utilizador final, em colaboração uns com os outros compõem a infraestrutura de *Cloud*.

No decorrer deste capítulo e do documento em geral, é dado um maior enfoque na *ótica* do administrador do sistema pois era um dos objetivos do projeto, dar a conhecer os serviços e as

interações entre estes para que, com base nessa informação, fosse possível desenhar e conceber uma estrutura sólida e flexível. No entanto foi também apresentado o ponto de vista do utilizador final sob a forma dos elementos que terá à disposição após a construção da *IaaS*, o que acontece no último tópico. No capítulo que se segue é concretizada a infraestrutura de *Cloud* que foi apresentada até ao momento, onde é detalhado o material utilizado, a arquitetura implementada e os serviços instalados em cada um dos elementos que irão constituir a solução.

6. DESENHO E IMPLEMENTAÇÃO DA INFRAESTRUTURA DE CLOUD BASEADA EM OPENSTACK

No capítulo anterior foi apresentada a arquitetura do *OpenStack*, os serviços e os elementos que os constituem bem com as interações existentes entre eles. Toda a informação apresentada serve de base para este capítulo, o qual apresenta a implementação prática de uma solução de *Cloud* baseada no *OpenStack*. Vai ser apresentado o *hardware*, delineada a arquitetura e identificados os elementos de *software* utilizados para dar vida à infraestrutura de *Cloud*.

Com tantas considerações e opções disponíveis, um dos objetivos principais é fornecer alguns caminhos claramente traçados e testados onde se possa ter uma visão clara que é viável implementar uma infraestrutura de *Cloud* de baixo custo, robusta, modular, escalável e ainda assim apresentar um bom desempenho mesmo com recurso a *hardware* de gama comercial mas, que este processo também sirva de base à exploração do *OpenStack*.

6.1. Considerações

Uma solução que é composta por vários projetos de *software*, faz dela modular e passível de se adaptar a diferentes cenários, no entanto, com tais benefícios vem a exigência no momento de decidir qual a melhor opção, qual a configuração a adotar face à realidade atual sem esquecer as mudanças de requisitos ou cenários a curto mas principalmente a médio prazo. Com isto em mente, pretende-se nesta secção dar o enquadramento de alguns pontos considerados importantes e quais as opções tomadas que ditaram o desenho e implementação da infraestrutura aqui apresentada.

6.1.1. Hardware e Dimensionamento

Ao desenhar uma solução de *Cloud* existe um objetivo traçado e uma necessidade identificado que, para dar resposta, é necessário identificar quantidades e qual o *hardware* a utilizar. A criação de máquinas virtuais dentro do *OpenStack* é feita com base em *templates* criados pelo administrador do sistema, onde são definidos parâmetros como memória *RAM*, *CPU* e tamanho do disco. É com base nestes valores, no número máximo expectável de *VM's* ativas em simultâneo e nos rácios de alocação entre o físico e o virtual, que se pode perceber qual a capacidade disponível com os recursos atuais ou por outro prisma, quantos mais recursos vou necessitar para atingir o patamar estabelecido.

Os rácios atribuídos por defeito pelo *OpenStack* são de 16:1 para o *CPU*, o que significa que por cada core físico iremos ter 16 cores virtuais, e 1.5:1 para o caso da memória *RAM*, ou seja aplica-se a seguinte fórmula:

$$\frac{(Rácio * Quantidade\ de\ recursos\ físicos)}{Quantidade\ de\ recursos\ virtuais\ por\ instância}$$

Numa plataforma onde existam 12 *cores* físicos, internamento o *scheduler* vai ter à disposição 192. Se cada instância utilizar 4 *cores*, então a plataforma conseguirá alocar 48 máquinas virtuais. Seguindo o mesmo raciocínio, se o servidor tiver um total de 48Gb de

memória *RAM* o sistema vai identificar 72Gb, assim, se cada instância utilizar 8Gb de memória, a plataforma só conseguirá alocar 9 máquinas virtuais.

Há ainda algumas perguntas que podem ser feitas e que vão decerto afetar a escolha dos componentes de *hardware*, quantidades e a arquitetura a adotar. Perguntas tais como:

- Quantas *VM's* se espera que corram em simultâneo?

- A plataforma vai ter de gerir os recursos de quantos *compute nodes*?

- Vai ter muitos utilizadores a aceder? Vai ser feito maioritariamente com recurso ao ambiente gráfico?

- Quantos serviços *nova-api* vão ser executados em simultâneo? (segundo a documentação o ideal é no máximo um por *core*)

- As *VM's* são muito ou pouco voláteis? No caso de haver muitos pedidos de criação e eliminação não é só o *Compute Node* que sofre mas também o *Controller* com todos os pedidos que chegam à sua *API* e a necessidade de os agendar e mandar executar.

Ao tentar encontrar uma respostas para alguns pontos-chave é possível elaborar uma estimativa para a quantidade de discos rígidos necessários, tendo como base o tamanho médio que o administrador do sistema pretende dar a cada instância, qual o débito a utilizar nas placas de rede, a capacidade de processamento e memória *RAM* ou eventualmente acrescentar um servidor para dividir os serviços de gestão da infraestrutura.

Relativamente ao espaço necessário para armazenamento das imagens que vão dar origens às máquinas virtuais, de discos virtuais para anexar às instâncias, e ficheiros, vai ser abordado de seguida no subcapítulo 6.1.4 dedicado para o efeito.

Quanto à escolha dos processadores, deve-se certificar que estes suportam virtualização.

6.1.2.Gestão do Risco e Escalabilidade

No que concerne a risco e escalabilidade, seja a curto ou médio prazo, são temas que devem ser equacionado e desenhados mesmo que na prática, seja por uma questão de falta de recursos ou pelo enquadramento do projeto, tais princípios não sejam colocados em produção. E sobre estes dois tópicos há algumas considerações que devem ser examinadas no momento de elaborar uma proposta de arquitetura de *Cloud* e de preferência colocadas em prática numa fase inicial do projeto de modo a evitar mudanças complicadas numa fase mais avançada ou mesmo a impossibilidade de alterar sem recomeçar do zero.

Qualquer sistema que envolva algum nível de criticidade surge sempre o tema da redundância e o que implementar para que tal seja possível. De entre muitas soluções a vários níveis da arquitetura, a implementação dos disco em *RAID* é uma delas. A construção do *RAID* é algo que deve ser planeado antecipadamente devendo ser feito em todos os elementos que vão integrar a solução de *Cloud*. Em caso de falha de um dos discos a recuperação do sistema é relativamente simples e rápida.

Uma outra situação algo mais dispendiosa mas que reduz ao máximo o tempo de indisponibilidade em caso de falha é existir a duplicação da entidade *Cloud Controller*, em que existe um em produção e outro em standby.

Até ao momento tem sido falado no *Cloud Controller* como sendo um só elemento, no entanto, qualquer serviço que seja da responsabilidade do elemento principal da *Cloud*, o *Cloud Controller*, pode ser separado em diferentes servidores para melhorar o desempenho e colmatar uma possível falha, ao mesmo tempo que deixa um maior leque de possibilidade de escalar a infraestrutura. No caso de se optar pela separação de serviço é de todo aconselhável a configuração de um balanceador de carga HTTP, seja por *hardware* ou *software*, devido à maior carga de comunicação que tal divisão vai colocar na infraestrutura.

Em termos do sistema de ficheiros a utilizar quando se vai instalar o sistema operativo, há alguma preferência por utilizar volumes lógicos, como por exemplo o *LVM*, no lugar do sistema de ficheiros simples do *Linux* devido à maior facilidade em adicionar um novo disco e agregar esse mesmo espaço à solução.

Relativamente à base de dados é usual utilizar um servidor exclusivo para o efeito no entanto, se se optar por uma arquitetura mais avançada para suporte de uma maior volume de máquinas virtuais, de equacionar-se utilizar uma implementação em Cluster.

Na altura que foi implementada esta solução o Neutron ainda não suportava uma configuração *multi-host* pelo que, se optou por utilizar o *Nova* para a gestão da rede da infraestrutura. Numa configuração *single-host* o serviço de rede está centralizado num servidor e como tal mais propenso a falhas e caso tal aconteça afeta toda a solução. No caso de ser implementada uma configuração *multi-host*, cada uma dos nós de computação irá correr um serviço de rede, neste caso o *nova-network*, assim, em caso de falha, só as *VM's* desse nó é que serão afetadas.

6.1.3. Rede

Há grandes vantagens em dividir os serviços e incluir mais máquinas para os alojar de modo a diminuir a carga de uma determinada máquina e minimizar o tempo inoperacional da plataforma em caso de falha, no entanto, estas opções vão incrementar a carga em toda a rede da solução.

Com isto em vista, em termos de rede, sempre que possível, deve-se optar por placas com o maior débito. Em implementações que os serviços de gestão da plataforma estão divididos por diferentes servidores ou que se decida armazenar os ficheiros de imagens em algum sistema de gestão de ficheiros, vai haver uma carga maior sobre a rede tanto na comunicação dos pedidos entre os serviços, como no carregamento da imagem no momento de criar uma máquina virtual. Em alguns casos pode-se optar por ter duas placas de rede ligadas entre si, pois desta forma a resposta aos pedidos pode ser dada por placas diferentes, dividindo a carga e diminuindo o tempo de espera.

6.1.4. Armazenamento

A complexidade do tema armazenamento está relacionada com o que se pretender implementar, e sobre isto existem quatro pontos que necessitam ser analisados para tomar uma

decisão informadas, nomeadamente: *i*) onde vão ser armazenadas as imagens para criação das *VM's*; *ii*) qual o espaço que vai ser preciso para as instâncias que estiverem a ser executadas; *iii*) se vamos querer ou precisar de implementar um sistema que forneça armazenamento de blocos/discos virtuais; *iv*) se vai ser preciso um sistema de armazenamento de ficheiros.

Os ficheiros de imagens *ISO*, que são utilizados para criar as máquinas virtuais da plataforma podem ser armazenados no próprio sistema de ficheiros do *Controller* ou de alguma implementação de armazenamento como é o caso do *Swift*. A plataforma permite ainda ir buscar imagens ao serviço *Amazon S3* ou descarregá-las via *HTTP* de um servidor *Web*. O espaço físico vai depender da diversidade das imagens que se pretendam disponibilizar na solução.

O próximo passo é definir quanto espaço será necessário para os discos que servem de base para as máquinas virtuais e onde esse espaço vai ficar na plataforma, se em separado do servidor de computação, o que por um lado vai precisar de menos *RAM* e *CPU* por outro coloca uma maior carga na rede, ou em conjunto no mesmo servidor de computação.

Se armazenarmos as instâncias em execução fora do servidor de computação facilita a manutenção dos servidores e em caso de falha as *VM's* são de fácil recuperação no entanto a comunicação pela rede pode ser uma desvantagem. Optando por utilizar o mesmo servidor para ambos os propósitos pode-se optar por um sistema de ficheiros distribuído, que, embora seja mais fácil de escalar no caso de se pretender adicionar armazenamento torna-se mais complexo de operar e recuperar *VM's* em caso de falha. Ainda com o mesmo servidor para a computação e armazenamento de instâncias em execução se não for utilizado um sistema de ficheiros distribuído, a sobrecarga de um determinado nó não afeta os restantes e o tempo de resposta pode diminuir devido a não depender tanto da comunicação em rede. Contudo em caso de falha de um servidor perdem-se as instâncias em execução e a migração de instâncias entre nós é mais complexa. Em relação ao espaço necessário, este vai depender do número máximo estimado de instâncias a serem executadas em simultâneo e o tamanho do disco que se pretenda atribuir a cada uma.

De seguida deve ser endereçada a questão do espaço de armazenamento extra que se pretenda atribuir à plataforma. Este espaço é transformado em blocos de armazenamento ou discos virtuais que podem ser anexados às máquinas virtuais assim que estejam em execução. Estes discos pertencem a uma máquina de cada vez mas podem ser anexados a várias, não perdendo a informação que lá foi guardada, ao contrário do que acontece com os discos que servem de base para as instâncias, que são eliminados ao mesmo tempo que a máquina virtual. A implementação desta funcionalidade está a cargo do *Cinder*. Na solução aqui implementada foi utilizado um disco de 250Gb anexado ao *Cloud Controller* a fim de testar o serviço pois esta solução não estava contemplada no propósito inicial, em parte, devido à volatilidade das *VM's* fruto da utilização para fins de testes e provas de conceito e não de trabalho continuado.

O último ponto referente às necessidades de armazenamento passa pelo *Swift*, serviço que presta uma solução de armazenamento de objetos ou ficheiros. Esta solução é bastante completa e com uma arquitetura algo intricada e que requer bastantes servidores e poder de computação, sendo que a arquitetura para a sua implementação de base prevê um mínimo de cinco servidores, um principal para gestão do serviço e os restantes para a replicação da informação entre si. Ao

ponderar pela implementação do *Swift*, devem ser consideradas placas de rede com o maior débito possível devido à carga expectável, seja devido à criação de novos objetos ou recuperação de uma falha, que implicam a replicação pelos vários nós do sistema. Esta solução não foi equacionada para este projeto devido à exigência em termos de *hardware* necessário para a sua implantação e porque a finalidade da *IaaS* assim não o exigia, aliado ao facto de a empresa já possuir outros sistemas do género.

6.1.5.Sistema Operativo

A escolha do sistema operativo é bastante importante na medida em que o tempo de suporte deste, a sua maturidade e disseminação da sua utilização pela comunidade pode ditar mais ou menos suporte no momento das configurações e dos elementos a integrar. Visto isto, a opção recaiu sobre o *Ubuntu Server 12.04 LTS 64 bits*. Optou-se pela versão *LTS*, *Long Term Support*, devido a existir durante um maior período de tempo suporte do sistema operativo em termos de atualizações de segurança, correção de erros ou *drivers*. A decisão pelos *64bits* deveu-se, em primeira instância, ao facto de tirar um maior proveito e gestão da totalidade de memória *RAM* instalada, que em todos os casos é superior a 4Gb; em segundo, para não condicionar a execução de máquinas virtuais exclusivamente com sistemas operativos de *32bits*.

Outro fator que deve pesar quando tem de se decidir sobre um determinado componente é o suporte que existe nas comunidades e, neste caso, a maior expressão recai sobre o *Ubuntu*, aliado ao facto de na empresa já existirem outras implementações tanto em *Ubuntu* como outros derivados do *Debian*, facilitou a decisão neste ponto.

6.1.6.Software OpenStack

Tal como aconteceu com o caso do sistema operativo, é necessário decidir por qual versão do *OpenStack* se deve optar e, na altura da implementação, a decisão recaiu sobre a versão estável que estava em vigor, e essa era o *OpenStack Havana*. Embora o ciclo de lançamento de novas versões seja bastante curto, seis meses, e o seu tempo de vida estimado superior a um ano, a escolha de versões consideradas estáveis ou finais é sempre de considerar como primeira opção.

Como a solução do *OpenStack* é um conjunto de projetos de *software* interligados entre si, o nível de maturidade de cada um deles difere. Exemplo disso é o caso do *Nova* e do *Swift*, que acompanham a solução desde a primeira versão, por outro lado temos o caso do *Heat* e do *Ceilometer* que foram lançados com a versão utilizada neste projeto. Por esse facto, o de serem projetos recentes e com pouca expressividade em implementações práticas, maturidade e documentação reduzida, se decidiu não os incluir na elaboração deste caso prático.

A escolha do *hypervisor* foi algo natural tendo em conta o sistema operativo, pois o *KVM* é o complemento mais comum com este sistema operativo e o mais adotado pela comunidade *OpenStack*. Visto existirem outras opções igualmente válidas, a escolha mais acertada será sempre por aquele que o utilizador melhor conhece e possa estar acostumado.

No caso do motor de base de dados, para além do fator comunidade *OpenStack*, documentação abundante e grande maturidade, a escolha foi influenciada pela utilização em projetos anteriores, fazendo do *MySQL* a opção mais sensata. Se ao nível da base de dados se

optar por uma arquitetura em *cluster*, descentralizando a base de dados por uma questão de ser mais tolerante a falhas, é aconselhável um outro motor também baseado em *MySQL* mas com maior suporte e integração para cluster, por exemplo o *MySQL/Galera*.

Muitos dos serviços da plataforma de *Cloud* comunicam entre si utilizando a fila de mensagens. Para esta implementação optou-se pela solução RabbitMQ por ser bastante fácil de implementar e usar. Embora soluções como o *Qpid*[2] ou *0mq*[3] sejam igualmente válidas e tenham algum suporte na comunidade, o *RabbitMQ* tem tido bastante implementação prática com bons resultados segundo a comunidade do *OpenStack*. Tem também suporte nativo para ser utilizado em *cluster*.

A compatibilidade do *OpenStack* com outros serviços é alargada ao *S3* e ao *EC2* da *Amazon*. Sobre este último é requerida uma pesquisa e leitura cuidada para coexistir com as *API's* internas e não gerar incompatibilidade e inconsistência na operação da plataforma. Isto deve-se ao facto de cada uma das *API's* ter uma maneira diferente de comunicar com as imagens e instâncias.

Uma outra tomada de decisão é requerida no momento de escolher como vai ser gerida a rede da infraestrutura. Neste caso as opções passam pelo serviço do *Nova*, o serviço que desde o início geria a rede, e o novo serviço *Neutron*. A decisão pelo serviço do *Nova* baseou-se em dois pontos-chave, primeiro, pelo motivo de poder utilizar uma configuração *multi-host*, descentralizando a gestão pelos vários nós o que minimiza a paragem total em caso de falha, em segundo, por se tratar de uma implementação relativamente pequena em termos de recursos. O servidor a ser utilizado pelo Neutron foi reutilizado para fornecer mais capacidade de computação para a solução final.

6.2. Arquitetura

O ideal sempre que se desenha uma solução é tentar encontrar um compromisso que permita encontrar a melhor relação custo/qualidade, no entanto, nem sempre se consegue atingir esse objetivo. Neste caso em concreto foi-nos disponibilizado um determinado conjunto de computadores e *hardware* adicional que levou a que fosse desenhada a melhor arquitetura de modo a tirar o máximo de proveito do equipamento.

Em termos macro, esta solução vai ter um *Cloud Controller*, com todos os serviços de gestão que lhes são usualmente associados e os restantes computadores vão servir de *Compute Nodes*, ou seja, vão fornecer toda a capacidade de computação da infraestrutura, desde memória *RAM*, processador e armazenamento. A título de resumo temos o seguinte:

[2] Solução que implementa filas de mensagens baseadas no protocolo AMQP (https://qpid.apache.org/)
[3] Solução que implementa filas de mensagens baseadas no protocolo AMQP (http://zeromq.org/)

Cloud Controller

- *Nova* – Computação + Rede
- *Glance* – Imagens
- *Cinder* – Discos virtuais
- *Keystone* – Autenticação
- *Horizon* – Interface gráfico
- Motor da base de dados – *MySQL*
- *RabbitMQ* – Fila de mensagens

Compute Node

- *Nova* – Computação + Rede
- *Hypervisor* – *Qemu/KVM*

Desde o capítulo anterior que tem sido apresentada a arquitetura do *OpenStack* e à medida que se vai avançando na dissertação vai sendo concretizada e sendo dado maior detalhe. De momento a arquitetura foi reduzida áquilo que são os projetos de *software* identificados para a implementação prática, de seguida, entramos no detalhe de cada um de modo a especificar quais os serviços necessários.

6.3. Pré-requisitos

Na instalação desta infraestrutura há alguns pré-requisitos a ter em conta, nomeadamente:

- Cada um dos servidores/máquinas deverá ter o sistema operativo instalado, de preferência a versão de *64bits*. Não é imperativo, no entanto, se o sistema operativo for de *32bits* não vai ser possível executar máquinas virtuais de *64bits*.
- Instalar a versão server da distribuição de *Linux* escolhida e sem ambiente gráfico.
- Cada um dos nós que integrem a solução deverão ter instalado o *Network Time Protocol* (*NTP*) de modo a que todos os equipamentos estejam assim sincronizados e os serviços tenham a mesma base de referência temporal.
- Deverá ser instalado o motor da base de dados que neste caso será o *MySQL*. No *Controller* vai ficar o servidor e o *Python* para que os serviços lhe possam aceder.
- Embora não seja obrigatório, pois a instalação funciona perfeitamente numa partição simples, é de todo aconselhável configurar partições lógicas (*LVM*) pois é mais prático de trabalhar o tamanho das partições ou mesmo adicionar mais espaço ao já existente.

6.4. Configuração de Hardware

Para facilitar a compreensão do exposto, doravante será utilizada neste documento a mesma terminologia que na documentação do OpenStack em que o termo *Cloud Controller* é utilizado para identificar o elemento principal na infraestrutura de *Cloud* e *Compute Nodes* para todos os outros, os elementos que fornecem a capacidade de computação da plataforma.

As principais características técnicas dos equipamentos utilizados são apresentadas na Tabela 6.1:

Tabela 6.1 - Características dos computadores utilizados na implementação da solução

	Cloud Controller	Compute Node
		
Marca:	HP Compaq	HP Compaq
Modelo:	8300 Elite SFF	8300 Elite SFF
Processador:	Intel® Core™ i3 @ 3.30GHz	Intel® Core™ i3 @ 3.30GHz
Memória:	8 Gb DDR3 / 1067 MHz (2x4Gb)	12 Gb DDR3 / 1067 MHz (3x4Gb)
Disco:	2 x 250Gb	250Gb
Placa de Rede:	2 x Gigabit Ethernet controller	2 x Gigabit Ethernet controller

Relativamente aos equipamentos utilizados é de salientar que são computadores de gama comercial em que a única alteração realizada foi a adição de memória *RAM* e de uma placa de rede *gigabit* em cada um. Para além dos computadores foram ainda necessários dois *Switches Layer 2 Gigabit Ethernet* com oito portas cada. A imagem que se segue representa os equipamentos utilizados e a topologia de rede adotada para este caso em concreto.

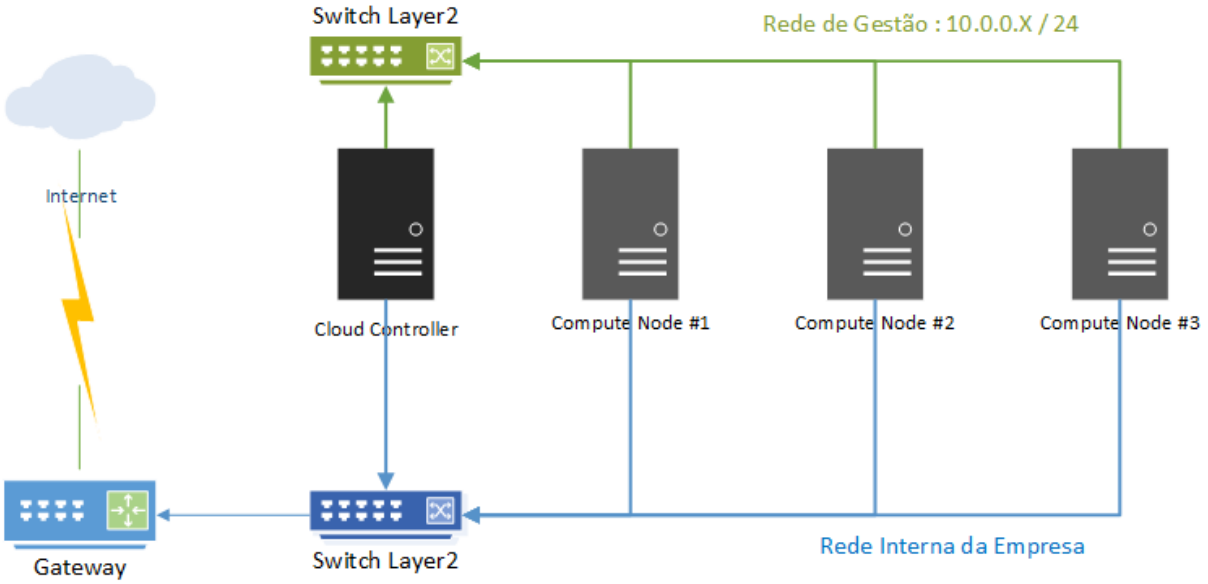

Figura 6.1 - Equipamentos e topologia de rede utilizados

Todos os nós da infraestrutura comunicam através de dois *switches*, um é utilizado para a comunicação interna das máquinas virtuais e o outro para garantir uma ligação com a rede interna da empresa. O *Controller Node*, embora assegure as principais tarefas de gestão da infraestrutura conjuntamente com a autenticação, rede, imagens, discos virtuais e o servidor Web para acesso ao interface gráfico, tem menos memória *RAM* que os restantes por uma questão de gestão dos recursos físicos que foram disponibilizados na altura para esta implementação.

As arquiteturas de *Cloud* exploram técnicas de virtualização para fornecer várias máquinas virtuais no mesmo servidor físico. A modularidade do *OpenStack* e diversidade de arquiteturas que são possíveis de desenhar e implementar faz com que seja possível gerir com maior eficácia os recursos disponíveis. Esta implementação não tem o serviço específico de rede, o Neutron, recorrendo para o efeito ao *Nova*, desta forma prescinde-se de um servidor. Do mesmo modo, não implementa o sistema de armazenamento de ficheiros, o *Swift*, que por si só, necessita de cinco servidores.

Com o desenho da arquitetura feito e com o conhecimento de todo o *hardware* disponível para a configuração da plataforma de *Cloud*, já nos é possível estimar a capacidade de recursos virtuais da solução final, de acordo com as fórmulas seguintes:

Total *vCPU's* $= [RacioCPU * (\#CoresFisicosPC * \#PCs)] = [16 * (4 * 3)] = 192$

Total *vRAM* $= [RacioRAM * (\#RAMFisicaPC * \#PCs)] = [1.5 * (12 * 3)] = 54\ Gb$

Total Disco $= (\#DiscoPC * \#PCs) = [250 * 3] = 750\ Gb$

Na nossa plataforma de *Cloud* vão ficar disponíveis 192 *cores* virtuais, 54Gb de memória *RAM* e 750Gb de espaço em disco para a execução de máquinas virtuais. Há agora que traduzir a capacidade adquirida em número de máquinas virtuais e, para isso, vamos recorrer aos valores médios de cada uma dos componentes definidos nos *templates* de criação de *VM's*, que na plataforma são chamados de *flavors*:

Se cada *VM* usar 4 *vCPU's*, vamos conseguir instanciar no máximo $(192/4) = 48$

Se cada *VM* usar 4Gb de *vRAM*, vamos conseguir instanciar no máximo $(54/4) \cong 13$

Se cada *VM* usar 60Gb de disco, vamos conseguir instanciar no máximo $(750/60) \cong 12$

Com os números acima apresentados podemos concluir que a plataforma tem capacidade para alocar no máximo 12 máquinas virtuais, no entanto este é um valor estimado e como tal pode variar consoante se aloquem mais ou menos de cada um dos recursos disponíveis.

Com esta configuração básica em que é utilizado *hardware* de gama comercial, o nosso objetivo de fornecer aos departamentos de TIC uma solução de *Cloud* onde se possam configurar rapidamente máquinas virtuais e dar as diretrizes para futuras implementações com vista à exploração da tecnologia de computação na *Cloud*, consegue ser atingido. Com as configurações e *hardware* atuais espera-se conseguir usufruir do triplo dos recursos físicos.

6.5. Configuração de Software

De seguida são descritos os componentes de *software* a serem instalados em cada máquina. A principal, que detém todas as atividades de gestão do ambiente, conhecido como *Cloud Controller* e os restantes nós ou satélites, denominados *Compute Nodes*, são as máquinas que suportam todos os recursos que ficarão disponíveis na infraestrutura.

6.5.1. Cloud Controller

A Figura 6.2 apresenta os projetos de *software* e todos os respetivos serviços que vão ser instalados no servidor principal.

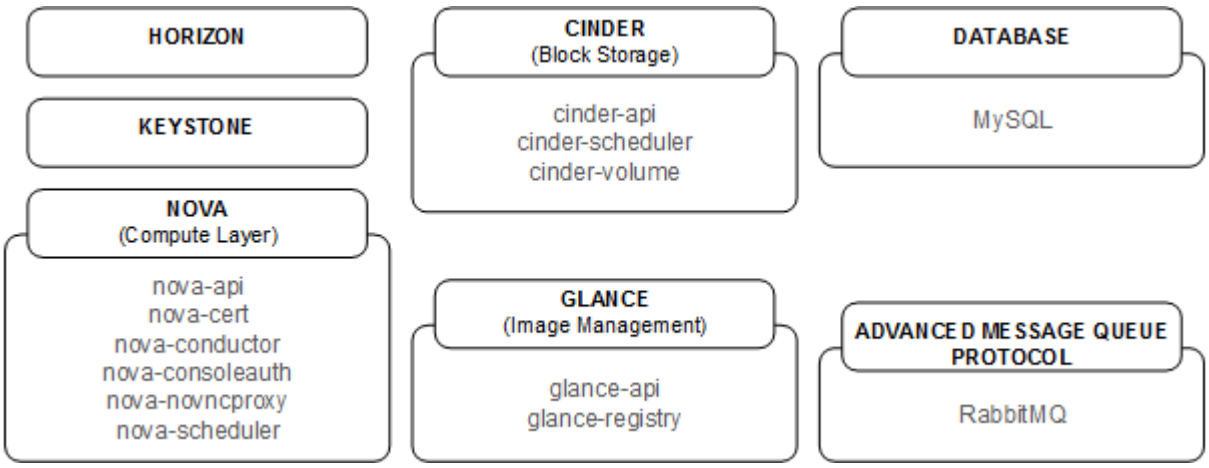

Figura 6.2 - Componentes de software a ser instalado no *Cloud Controller*

Como servidor principal da solução de *Cloud*, este vai desempenhar as seguintes funções:

- Gestão da Computação com recurso ao *Nova*.
- Autenticação e autorização através do *Keystone*.
- Servidor web que terá alojado os componentes do interface gráfico usando o *Horizon*.
- A gestão das imagens que servem de base para as *VM's* utilizando o *Glance*.
- Gestão de discos virtuais ou blocos de armazenamento via *Cinder*.
- Motor de base de dados *MySQL*.
- Servidor do *RabbitMQ* para a implementação da fila de mensagens.

Nova é o componente de *software* principal da plataforma. No servidor principal vai ser instalado o *nova-api*, componente que recebe e faz a distribuição dos pedidos para os seus serviços internos e o *nova-cert* que faz a gestão dos certificados da plataforma. O *nova-novncproxy* é o componente de *software* que permite o acesso às instâncias através de clientes *VNC* que fazem a ponte entre rede pública e rede interna em que as conexões são autorizadas mediante validação do *token* do utilizador, que é fornecido pelo *nova-consoleauth*. O *nova-scheduler* gere os pedidos de recursos feitos pelos utilizadores e seleciona o nó de computação mais adequado para as instâncias serem executadas. Os critérios de seleção do servidor consideram fatores como memória física disponível, armazenamento, processador ou carga de

rede, sendo que, por defeito, a escolha recai no servidor que tenha menor carga ou mais recursos disponíveis.

O *Glance* é o serviço que gere as imagens da plataforma; ele fornece serviços de descoberta, registo e entrega de imagens, capaz de copiar ou capturar instantaneamente uma imagem de uma instância e armazená-la ou usá-la mais tarde como um modelo para a criação de novos servidores. Esses recursos são fornecidos pelo *glance-api*, o ponto focal onde são concentrados os pedidos de imagens e o *glance-registry* para processar meta dados de imagens como o seu tamanho ou tipo.

Para existir a possibilidade de ter discos virtuais na plataforma de IaaS foi instalado o *Cinder*, nomeadamente a *API* para receber os pedidos de blocos de armazenamento, o *cinder-volume* para executar esses pedidos e manter atualizada a informação na base de dados e por fim o *cinder-scheduler* para determinar qual o local onde vai ser executado o serviço.

Nesta configuração, como em todas as que se pretendam implementar, são necessários os serviços considerados transversais a toda a infraestrutura. O *Horizon*, que facilita a interação dos utilizadores com a plataforma de *Cloud* ao fornecer um interface gráfico simples e completo, por outro lado o *Keystone*, que proporciona segurança através do seu serviço de autenticação.

Semelhante ao *Horizon* e *Keystone*, cada implementação do *Openstack* precisa de um servidor de base de dados e uma fila de mensagens. Por padrão e nesta implementação foi utilizado o *MySQL* como motor da base de dados e *RabbitMQ* como o sistema de mensagens entre serviços.

6.5.2.Compute Node

A Figura 6.3 apresenta os projetos de *software* e todos os respetivos serviços que vão ser instalados em cada um dos restantes nós do sistema.

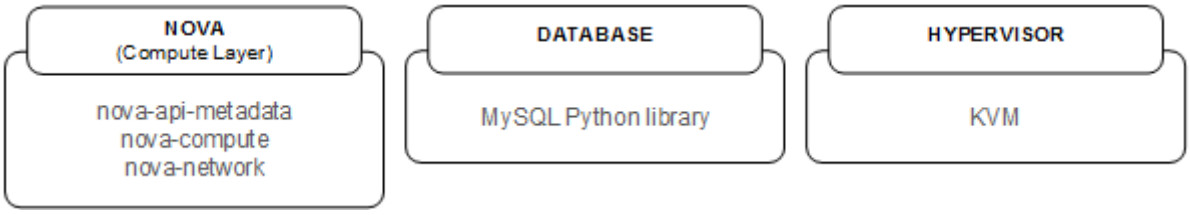

Figura 6.3 - Componentes de *software* a ser instalado nos *Compute Nodes*

Os nós de computação são as máquinas responsáveis por entregar aos utilizadores finais os recursos disponíveis na plataforma de *Cloud* sob a forma de máquinas virtuais ou armazenamento. Eles executam a componente de virtualização, onde utilizam como *hypervisor* o *KVM*. Além disso disponibilizam serviços de rede e implementam grupos de segurança através do uso do *nova-compute* e *nova-network*. Sendo que o motor da base de dados está instalado no servidor principal, cada um dos nós que seja adicionado à infraestrutura só vai necessitar do *Python* para que os serviços consigam aceder ao *MySQL*. O *Nova* não tem um *hypervisor* específico implementado; em vez disso, tem uma camada de abstração que permite que os utilizadores escolham de entre alguns fornecedores conhecidos.

6.6. Administração da plataforma

No final de conhecer a arquitetura dos serviços, ter desenhado o plano de ação e após tudo instalado é agora possível aceder ao ambiente gráfico da plataforma de *Cloud*, tirando proveito de todo o conhecimento adquirido. A solução final pode ser percecionada de três maneiras diferentes, consoante os privilégios atribuídos ao utilizador, o administrador do sistema, que instala e gere todo o ambiente, o utilizador e administrador do projeto que é criado dentro do OpenStack e o utilizador final que usufrui do acesso ás máquinas virtuais.

O utilizador final só irá tirar partido pela utilização de uma máquina virtual, seja através de uma ligação por *VNC* ou através de um terminal *dummy* ou *VDI*, conforme pode ser visto no subcapítulo 6.7 onde foram colocadas duas máquinas virtuais em pré-produção.

O utilizador membro, ou seja, sem privilégios de administração sobre a plataforma, ao ser criado e adicionado a um projeto – área de trabalho que tem à disposição um determinado conjunto de recursos atribuídos pelo administrador da plataforma – sobre o qual pode gerir os recursos cedidos. Na Figura 6.4, exemplo da página de entrada de um dos projetos criado nesta implementação, os utilizadores podem criar 4 máquinas virtuais e têm à disposição 50 *CPU's* virtuais, 50Gb de memória *RAM* e 10 endereços *IP* para que seja possível aceder às instâncias do exterior. As ações de gestão do membro de um projeto passam pela criação de máquinas virtuais, criar e atribuir discos virtuais, volumes, criar novas imagens e colocá-las como públicas para uso geral ou privadas ao projeto em causa e por fim a criação de grupos e regras de segurança.

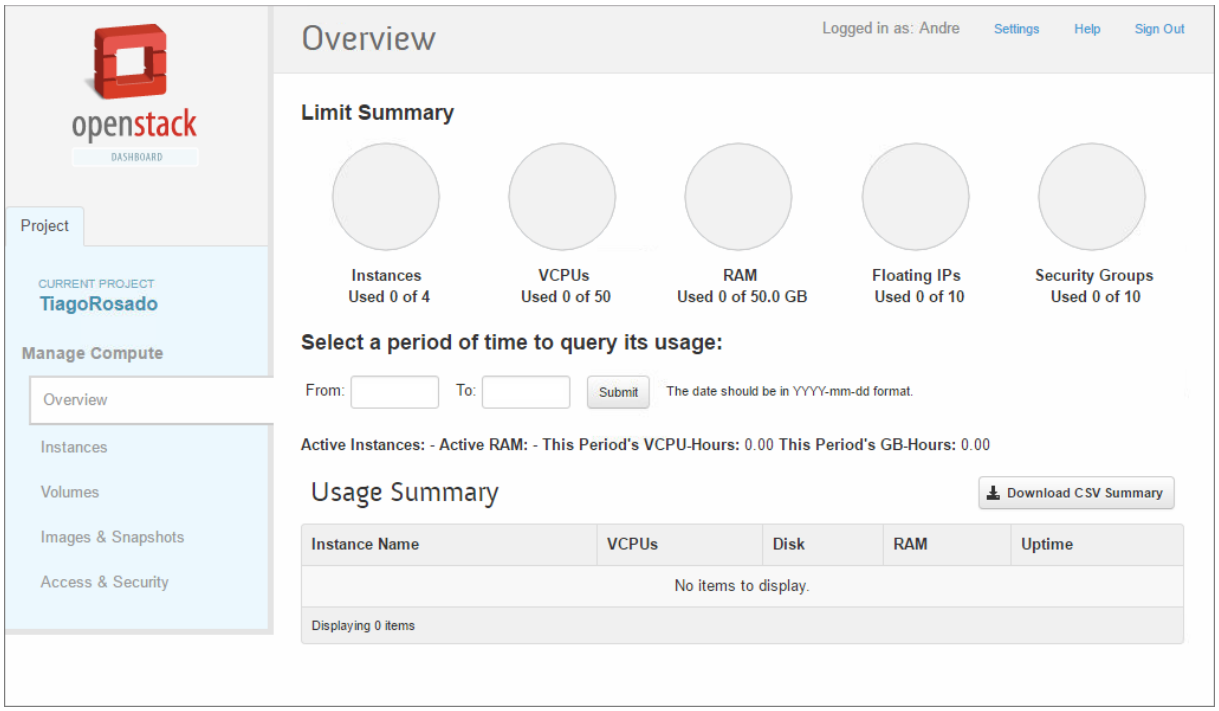

Figura 6.4 - Ambiente de trabalho de um membro da solução de *Cloud*

O espectro de ação do administrador do sistema é bem mais alargado. Ele tem a possibilidade de gerir a globalidade de recursos afetos à solução de *Cloud* e perceber como estes se encontram distribuídos, como por exemplo quais os recursos que estão em produção e em qual projeto. A Figura 6.5 é exemplo da vista de um administrador do sistema sobre todos os recursos disponíveis e qual a alocação de cada um dos *Compute Nodes*.

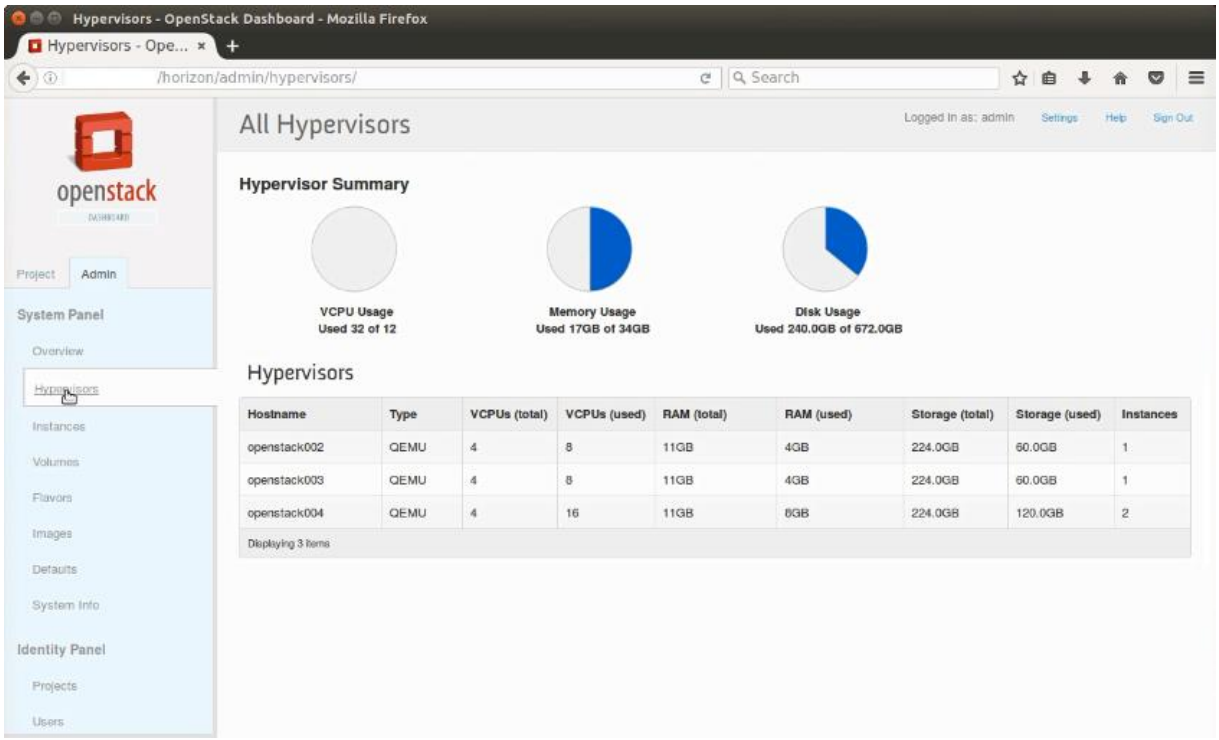

Figura 6.5 - Visão geral da alocação de recursos da plataforma de *Cloud*

De todas as tarefas possíveis de realizar sobre a plataforma, a de maior foco passa pela criação de uma máquina virtual. A Figura 6.6 mostra isso mesmo, os passos necessários para a criação de uma instância. Do lado direito são apresentados gráficos indicativos dos recursos que ainda se encontram disponíveis e os recursos do *flavor* escolhido, do lado esquerdo podemos dar nome, escolher o *flavor* adequado e qual a imagem de base.

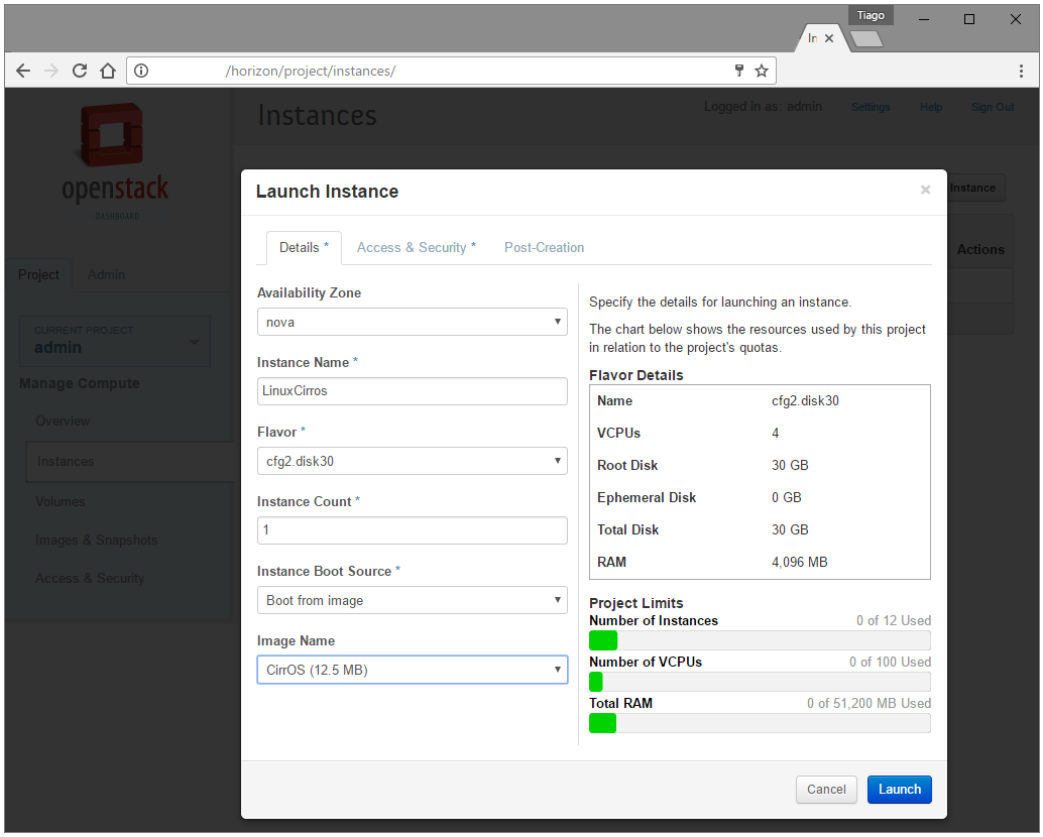

Figura 6.6 - Menu de criação de uma instância no *OpenStack*

6.7. Caso em ambiente real

Como teste de carga à plataforma e prova de que a solução é viável em ambiente real, foram colocadas duas máquinas virtuais em produção. Na secretária do utilizador final foi colocada uma *VDI* que por sua vez tinha sido configurada para se ligar ao endereço *IP* de uma máquina virtual para que este estivesse a trabalhar como se de um computador físico se tratasse.

À data da instalação da plataforma a estação padrão que se encontrava em produção era baseada no sistema operativo *Windows XP*, tendo sido a que nos foi fornecida para os testes.

Figura 6.7 - Implementação em ambiente real

O posto de trabalho que se pode ver na imagem é um dos locais onde é feito o atendimento da linha do *Helpdesk* interno da empresa, local gentilmente colocado à disposição para que fosse possível implementar o ambiente de testes. À direita na imagem pode ser visto uma *VDI*, equipamento da *Wyse* que normalmente é utilizado com infraestrutura Citrix mas que neste caso foi reconfigurado para se ligar ao *IP* de uma máquina virtual alojada na plataforma de *Cloud* do *OpenStack*.

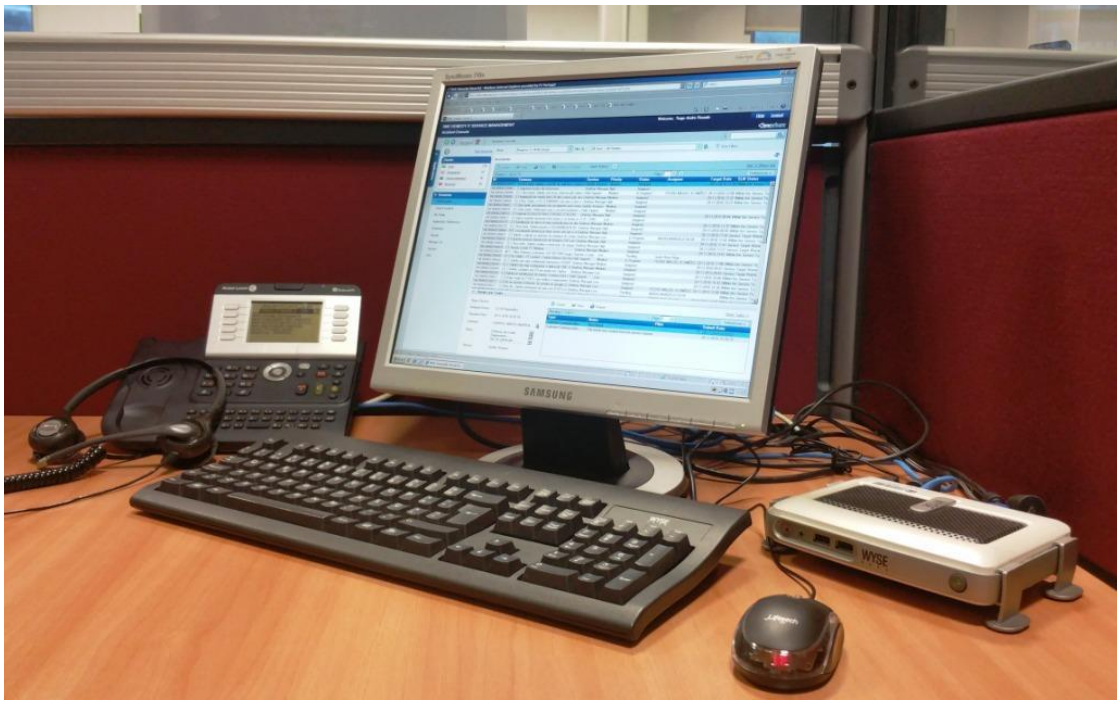

Figura 6.8 - Posto de trabalho sobre uma *VM* implantada na plataforma do *OpenStack*

A imagem que foi fornecida na altura para a realização deste teste era a "Estação Padrão" utilizada na altura e vinha com o sistema operativo *Windows XP* para além de algum *software* pré-instalado como por exemplo o *Office 2007*, *Internet Explorer* e *Google Chrome*.

Para o *Windows* funcionar numa máquina física ele utiliza os drivers *IDE* ou *SATA* que já estão incluídos na própria imagem do sistema operativo, no entanto, para trabalhar numa máquina virtual, e neste caso mais em concreto com o *hypervisor KVM*, é necessário instalar os drivers *virtIO*, disponíveis *online* em (*Windows VirtIO Drivers, 2014*). Estes *drivers* virtuais fazem a ponte entre os componentes, como por exemplo, os controladores do disco e da placa de rede, com o *hypervisor* assegurando desta forma um maior desempenho da máquina virtual.

Para esta situação, em que a imagem já estava feita e pronta para ser instalada em máquinas físicas, houve a necessidade de a alterar para incluir os drivers virtIO. Estes drivers são então adicionados ao ficheiro *ISO* da imagem, neste caso do *Windows XP*, para que o ficheiro possa ser carregado na plataforma (*Windows guests, 2014*).

Depois de criada a máquina virtual foi-lhe atribuído um *IP* público, neste caso um *IP* da pool de *IP's* da rede interna. Quando são criados projetos dentro da plataforma do *OpenStack* há a possibilidade de configurar a quantidade de *IP's* que esse projeto terá à disposição.

Estes *IP's* são ditos públicos pois são o meio que existe à disposição das máquinas virtuais para comunicarem com o exterior da plataforma. Como esta se encontra inserida na rede da empresa, vai usufruir do endereçamento disponível internamente, sendo a gestão da quantidade de *IP's* a atribuir a cada projeto da responsabilidade do administrador do sistema.

Para ser possível aceder à máquina virtual que se encontrava criada na plataforma de *Cloud* do *OpenStack* foi necessário aceder às configurações da *VDI* e especificar qual o local onde o terminal se vai ligar, que neste caso em concreto foi indicado o IP público atribuído anteriormente à *VM*.

Desta forma foi-nos possível ter duas máquinas virtuais a funcionar em produção para criar este ambiente de testes.

7. AVALIAÇÃO EXPERIMENTAL

No capítulo anterior apresentou-se o conceito de *Cloud Computing*, como se organiza e de que forma é apresentada ao utilizador final. Foi ainda detalhada uma plataforma *open source* de *Cloud – OpenStack –* e respetiva arquitetura implementada e colocada em produção. Como complemento a essa implementação foram realizados testes de performance, focados no tempo de *boot* das máquinas virtuais.

Neste capítulo é apresentado o estado da arte sobre testes de performance realizados em *benchmarks* de ambientes *IaaS*. Por fim, são ainda apresentados os dados que foram recolhidos na plataforma implementada com recurso ao *software Rally* (*Rally, 2015*), uma ferramenta de *benchmarking* e um dos projetos do *OpenStack* que complementa a sua oferta de soluções de *Cloud*.

7.1. Testes de performance: estado da arte

O *Cloud Computing* está a crescer continuamente como uma tecnologia importante para as empresas (*CA technologies, 2013*). Embora exista a competição entre os grandes fornecedores de serviços de *Cloud* pública, como a *Amazon*, *Microsoft*, *IBM* ou a *Google*, pelo aumento de quota de mercado, ainda existem empresas com algumas preocupações relativamente à privacidade e controlo destas soluções (*CA technologies, 2013*), evidenciando desta forma a necessidade de ter soluções de *Cloud* privadas.

Dos muitos estudos que existem sobre o tema em causa, são várias as abordagens feitas no que respeita à análise e comparação de plataformas de *Cloud Computing*, conforme será enunciado e analisado de seguida. Dos estudos identificados, há os que abordam estas soluções pela análise das suas características de mais alto nível, como a escalabilidade, elasticidade, facilidade de instalação ou gestão dos seus módulos constituintes, fornecendo assim uma boa base de conhecimento sobre as plataformas *open source* de *Cloud Computing* por forma a estabelecer um ponto de partida no momento de decidir qual a solução a adotar. Uma leitura mais aprofundada leva-nos a autores que dedicam o seu estudo a componentes específicos de modo a conseguir medir e avaliar as plataformas em termos de robustez, tolerância a falhas, escalabilidade ou performance.

Uma comparação entre plataformas de *Cloud* é essencial para conhecer as características e os pontos fortes de cada uma em maior detalhe. O estudo apresentado por (*Barkat, 2014*) elucida-nos sobre os três modelos principais, *SaaS*, *PaaS* e *IaaS*, sendo este último onde se enquadra o foco do seu trabalho, o estudo das características gerais da arquitetura, as propriedades mais importantes e a as funcionalidades que o *Openstack* (*Openstack, 2014*) e o *Cloudstack* (*Cloudstack, 2016*) disponibilizam. A arquitetura do *Openstack* é descrita como estando dividida em três grupos, Computação – a cargo do módulo de seu nome Nova – *Network* – atualmente com um módulo dedicado, o *Neutron* – e o Armazenamento – a cargo do componente *Cinder* ou *Swift*, dependendo do modelo a implementar. Por outro lado, o *Cloudstack* tem a sua arquitetura subdividida em 8 componentes, nomeadamente *Host*, *Cluster*,

Pod, *Zone*, *Region*, *Management Server*, *Storage* e *Networking*, sendo que no seu ambiente minimalista não são todos instalados ou configurados. Este estudo conclui que tanto em termos de funcionalidades – hypervisors suportados, ambiente de administração e de monitorização – bem como propriedades – balanceamento de carga, tolerância a falhas ou compatibilidade com outras plataformas de Cloud – são idênticos, tornando esta uma distinção muito difícil de fazer.

Do mesmo modo (*Bist, 2014*) apresenta uma análise detalhada visando somente as características das plataformas *open source* de *Cloud Compuntig* como por exemplo o ano de início do projeto, hypervisors suportados, compatibilidade com outras plataformas, armazenamentos suportados ou linguagem em que foi desenvolvida a plataforma. Desta feita o estudo incidiu sobre a *Deltacloud* (*Deltacloud, 2015*), o *Openstack* (*Openstack, 2014*) e o *Xen Cloud Platform* (*Xen Cloud Platform, 2015*). O objetivo do *Delta Cloud* passa pela implementação de uma camada de abstração com a finalidade de disponibilizar uma interface única para a gestão de várias infraestruturas de *Cloud Computing*, tais como a *Amazon EC2*, *GoGrid*, *OpenNebula* e *Rackspace*, permitindo assim o usufruto das características de diferentes infraestruturas. A plataforma *XenCloud* em termos de características é em tudo semelhante, sendo o ponto de divergência mais notório a disponibilização de suporte a um único *hypervisor*, o *Xen*. Foi concluído desta forma a comparação entre três das plataformas com que se podem criar infraestruturas de *Cloud Computing*.

Em mais um estudo comparativo, (*Laszewski, 2012*) analisa quatro grandes plataformas *open source* que se propõem para a construção de infraestruturas de *Cloud Computing*. Este estudo, para além da análise comparativa das características do *OpenStack* (*Openstack, 2014*), do *Eucalyptus 2.0* (*Eucalyptus, 2014*), do *Nimbus* (*Nimbus, 2015*) e do *OpenNebula* (*OpenNebula, 2015*), introduz o tema da escalabilidade e da concorrência conduzido pelo teste de arranque de várias instâncias em simultâneo em cada uma das frameworks em estudo. Estes testes foram feitos com recurso ao projeto *FutureGrid*, atualmente chamado de *FutureSystems* (*FutureSystems, 2015*), que agrega poder de computação para fins educacionais e de investigação, e que o torna um ponto de entrada para as diferentes infraestruturas de *Cloud*. Conseguiu-se identificar problemas de escalabilidade, com maior notoriedade, no *Eucalyptus* e no *Openstsck*, em que aumentando a concorrência de *VM's* para 16 se obtinham taxas de erro de 50% no arranque das mesmas, por outro lado, está documentado na comunidade *Openstack* que a variedade e complexidade dos serviços instalados deverão aumentar de modo a fazer face ao aumento da exigência feita á plataforma. Neste estudo a plataforma *OpenNebula* foi vista como confiável e de fácil instalação no entanto verificou-se ser mais lenta na disponibilização das instâncias (*VM's*).

Concluiu-se que a adoção de uma plataforma de *IaaS* vai depender dos requisitos das aplicações e da comunidade de utilizadores sendo que o *OpenNebula* e o *Nimbus* eram de mais fácil instalação por outro lado *Eucalyptus* e o *Opensatck* tinham melhorado nas mais recentes versões tornando-os mais populares na comunidade de investigação ligada ao *FutureGrid*.

Estudos como o de (*Gillam, 2014*) apresentam uma análise de alguns fatores de risco em que foram realizados testes de largura de banda da memória *RAM* com o *STREAM* (*STREAM, 2014*), medida a performance das operações de leitura e escrita dos discos com o *Bonnie++* (*Using Bonnie++, 2014*) e operações aritméticas de vírgula flutuante com recurso ao *Linpack*

(*Dongarra, 2001*). Esses tinham a finalidade de mostrar que os níveis de performance das infraestruturas são importantes e devem ser tidos em consideração para a definição dos níveis de serviço a acordar com os fornecedores de soluções de *Cloud* públicas no momento da realização de um contrato de prestação de serviços. Embora estejam em foco soluções proprietárias, nomeadamente, da *Amazon*, *Rackspace* e *IBM*, é incluído nos testes uma solução de *Cloud* privada do *Openstack*.

O estudo específico que incide na performance dos hypervisores, realizado por (*Jayasinghe et al. 2013*), revela um trabalho aprofundado com incidência na performance em termos de taxas de transferência e latência nas respostas – estas medidas foram avaliadas com recurso ao *RUBBos* (*RUBBoS, 2014*) e ao *Cloudstone* (*Sobel, 2008*) – de *Clouds* públicas para a avaliar a viabilidade da migração de aplicações multicamada. Por forma a validar os testes realizados sobre infraestruturas de *Cloud* públicas, foram realizados os mesmos testes sobre os conhecidos hypervisores *Xen*, *KVM* e *CVM* (este último de cariz comercial) com recurso a implementações de *Clouds* privadas onde foram corroborados os resultados anteriores, validando o estudo realizado por (*Gillam, 2014*), em que testes realizados para o mesmo fornecedor de serviços Cloud mas para localizações geográficas diferentes, obtiveram resultados de performance diferentes. Verificou-se que a configuração de *Cloud* com melhores resultados de performance para o prestador *Emulab* (*Emulab, 2015*) resultou no inverso para a *Amazon*. Em termos da avaliação dos hypervisores, o *Xen* supera o hypervisor comercial em 75% no que respeita aos testes de leitura/escrita feitos com recurso ao *RUBBoS* e por sua vez o hypervisor comercial supera em mais de 10% nos testes de carga realizados sobre a plataforma *Cloudstone*.

Os testes realizados por (*Borhani, 2014*), apresentam uma abordagem sistemática de avaliação de desempenho de aplicações com uso intensivo de bases de dados em *Clouds* públicas. Embora tenha sido utilizada uma implementação de *Cloud* privada em *Openstack* de modo a criar testes de carga e de concorrência, o intuito foi testar a resposta das bases de dados em ambientes de nuvem pública e, como complemento, perceber o comportamento do *CPU* das máquinas virtuais. Os testes de carga realizados, do tipo leitura, escrita e leitura/escrita, revelaram que, em média, havia melhores tempos de resposta por parte das máquinas virtuais da *Rackspace* com configurações de memória, disco e *CPU*, mais pequenas. Para configurações de maior capacidade a *Rackspace* só conseguiu melhores tempos de resposta para as operações de Leitura, tendo a *Azure* obtido melhores prestações para as operações de Escrita e Leitura/Escrita.

O estudo levado a cabo por (*Xu, 2014*) visa a performance do componente do *Openstack* responsável pela gestão da rede, denominado de *Quantum*. Para este teste de performance da rede foram implementadas duas configurações, ambas com vários servidores, onde numa delas os serviços de rede estão centralizados e na outra estão distribuídos pelos vários servidores que compõem a infraestrutura. Para realizar os testes de carga na rede recorreu-se a operações de *Map* e *Reduce* da plataforma de computação distribuída *Hadoop* (*Hadoop, 2015*), em que a sua implementação num conjunto de 10 máquinas virtuais no mesmo servidor físico serviu de base de comparação para os testes realizados para a mesma infraestrutura *Hadopp* distribuída por 10 máquinas virtuais em servidores separados para o primeiro e segundo cenários acima identificados. Os resultados seguiram o esperado, em que houve uma performance maior para

a computação distribuída por vários servidores face à base de referência, no entanto a diferença de performance entre os dois casos em teste revelaram-se mínimas, inferior a 60 segundos para a mesma quantidade de *gigabytes* processados. Este estudo ajudou a mostrar que, embora o aumento de performance seja mínimo, é preferível distribuir os serviços de rede pelos vários servidores de modo a aumentar a fiabilidade do sistema em caso de falha.

A plataforma de testes criada por (*Ge, 2014*) automatiza testes de performance segundo três vertentes, operações realizadas por uma aplicação em Java, testes de carga a um servidor *Web* e quantidade de transações a uma base de dados transacional. A validação deste conjunto de testes a plataformas de *Cloud* teve por base uma infraestrutura de *Cloud* privada baseada no *Openstack* e uma pública, a *Amazon*. Relativamente aos testes de carga com recurso a uma aplicação em Java para a realização de cálculos, revela um aumento de performance de acordo com o aumento de memória das instâncias, o que se verificou para as duas infraestruturas, no entanto, o *Openstck* obteve melhores resultados devido á utilização de processadores mais recentes e do rácio de 1 núcleo virtual para 1 núcleo real. Para o caso da performance de rede, testado com pedidos realizados a um servidor Web, obtiveram-se novamente melhores resultados para o *Openstcak*, em parte, devido às restrições colocadas pela *Amazon* na atribuição de largura de banda de acordo com as características das instâncias, sendo que, para a maior obteve-se um máximo de *1Gbits/s*, 10 vezes inferior ao *Openstack* devido ao único limite ser a velocidades das placas de rede. Para os testes realizados com transações de Base de dados o *Openstack* evidenciou um aumento de transações com o aumento do número de *VCPU's* das instâncias, o que não foi verificado na *Amazon*, onde a variação do número de transações pouco variou com o tamanho das instâncias, o que levou os autores a atribuir esta situação à largura de banda dos volumes onde estão a correr as máquinas virtuais.

O estudo realizado por (*Plugaru, 2014*) para a combinação da plataforma de *IaaS* do *Openstack* com os processadores de baixo consumo energético *ARM* ou *Atom* para processamento ao nível de Computação de Alta Performance (*HPC*), verificou uma quebra de 24% em performance e cerca de 65% na capacidade de comunicação em comparação com ambientes não virtualizados.

Este estudo, focado na análise e validação da plataforma de testes de performance desenvolvida pelos autores, embora tenha obtido resultados pouco eficientes em termos energéticos e de desempenho, devido em muito pela fase inicial em que se encontra o desenvolvimento deste tipo de processadores para soluções de virtualização, denota que começa a ser possível ter alguma capacidade de computação em plataformas de *Cloud* compostas por processadores de baixo consumo energético.

Em outro estudo focado na memória, processador, capacidade de computação e armazenamento, aparece o trabalho de (*Varghese, 2014*) que aplica testes de carga realizados através de uma aplicação de cálculo de risco financeiro e outra de simulação de alterações moleculares.

Num teste comparativo entre duas plataformas de *Cloud* privada realizado por (*Qevani, 2014*) em que apresenta um teste qualitativo das características do *Openstack* e do *Synnefo*, aborda também aspetos quantitativos da performance, apresentados sob a forma de testes de

arranque de máquinas virtuais explorando a concorrência de pedidos. Este estudo incide no tempo decorrido desde o pedido da máquina virtual até ao momento em que esta fica disponível para ligações SSH e o tempo necessário para a eliminar. Para um total de 10 máquinas virtuais, os autores levaram duas abordagens em consideração, a primeira, onde foram feitos quatro incrementos até perfazer as 10 *VM's* o tempo máximo não excedeu os 60 segundos, na segunda abordagem, onde foram pedidas as 10 *VM's* em simultâneo, o tempo máximo não excedeu os 250 segundos. Com estes testes verificaram-se tempos de resposta maiores no *Openstack* quando são feitos pedidos de muitas máquinas virtuais em simultâneo.

Uma análise simples e sucinta foi dirigida por (*Steinmetz, 2012*) em que medem o tempo de lançamento – tempo decorrido desde o pedido até que a mesma fica pronta a ser acedida – com incrementos de uma máquina virtual nas plataformas *Openstack* e *Eucalyptus*. Este estudo, com uma abordagem semelhante ao de (*Qevani, 2014*) revelou que o *Openstack* demonstra uma melhor performance para pedidos unitários com um pico máximo de 6 segundos face aos 12 verificados no *Eucalyptus*. Ao contrário, quando no mesmo pedido são solicitadas várias máquinas virtuais, obtiveram-se máximos de 12 segundos para 4 máquinas em simultâneo, face aos 8 segundos registados no *Eucalyptus*, plataforma onde se verificou que á medida que se foi aumentando o paralelismo houve uma diminuição do tempo de resposta, ou seja, onde uma máquina demora 12 segundo a ficar disponível, 4 máquinas demoraram 8 segundos, comportamento contrário ao que acontece no *Openstack*, e que se deve ao facto de como cada umas das plataformas gerem os pedidos internamente.

Na análise realizada por (Paradowski, 2014) são feitos testes aos tempos de resposta da infraestrutura quando são realizados pedidos de criação e eliminação de instâncias – máquinas virtuais que foram parametrizadas com diversos valores para tamanho de disco, memória ou CPU – em ambos os ambientes de Cloud, Openstack (Openstack, 2014) e Cloudstack (Cloudstack, 2016). Os resultados são apresentados sob a forma de gráficos onde é feita a análise aos tempos despendidos para a realização de cada uma das operações e qual o impacto do tamanho do disco e da combinação de memória/CPU na performance da infraestrutura que dá suporte à solução de Cloud. Como resultado da sua análise verificou-se que existe uma forte correlação entre o tamanho do disco e o tempo de boot, bem como com o tempo de delete, tendo sido este o único parâmetro passível de ser identificado como tendo impacto na performance. Embora não tão expressiva contudo verificou-se que existe uma correlação entre o conjunto memória / CPU e o tempo de boot, o mesmo já não se verificou para o tempo de delete das VM's. Com estas conclusões foi colocado á prova se o tamanho do disco causava algum impacto na performance da máquina física, o que acabou por não se verificar. Em termo de conclusão da análise deste estudo e fazendo ele uma avaliação comparativa com a Plataforma da CloudStack, verificou-se que, em todos os casos de estudo o OpenStack revelou melhores prestações nos tempos medidos.

A diversidade existe, são várias as soluções apresentadas, os objetivos demonstrados, hipóteses deixadas em aberto para testes futuros e, no entanto, uma lacuna no que concerne, não à obtenção dos dados mas, às ferramentas utilizadas para o efeito e de que forma foi possível extrair esta informação. A análise que apresentamos fez a recolha dos dados para estudo com recurso a uma ferramenta que se acredita ser fiável e com total integração com a infraestrutura

de Cloud do OpenStack visto ser parte integrante da equipa de desenvolvimento da própria plataforma. Com vista a diferenciar-se dos estudos anteriores foram introduzidas novas variáveis, como o sistema operativo, e avaliados os componentes em separado, memória, disco e CPU. Aliado a estas alterações foi introduzido o fator concorrência, ou seja para a mesma flutuação de valores e sistema operativo, foram testados cenários com cargas diferentes, em que os pedidos contemplavam duas, quatro ou seis máquinas virtuais em simultâneo.

7.2. Plataforma Rally

O *OpenStack* é descrito em toda a literatura como uma infraestrutura robusta e altamente modular, a mesma que foi apresentada nesta dissertação, tendo sido explorados os principais projetos de *software* que, alguns por si só representam uma funcionalidade, ou que como um todo e devidamente configurados e interligados formam uma plataforma de *Cloud Computing*. A sua grande variedade de configurações, tanto de *software* como de *hardware* em que o *OpenStack* pode ser implantado, leva a que, para tentar testar e entender qual o efeito de alterações à infraestrutura ou de determinadas decisões de implementação no desempenho e comportamento da solução final seja preciso lidar com a sua complexidade.

O projeto *Rally*, embora não seja um elemento fundamental para a criação de uma solução de *Cloud*, é parte integrante do *OpenStack*. É um projeto bastante recente, do início de 2015, que nasceu da necessidade de obter informação para indicadores de desempenho e qualidade dos serviços existentes e soluções implementadas.

Rally é uma ferramenta de *benchmarking* que coloca à prova cada um dos serviços da solução ou a plataforma como um todo através da execução de testes. Estes testes de carga podem ser executados individualmente ou conjugados entre si por forma a criar os cenários que consigam responder à avaliação pretendida ou validar uma determinada situação, tentando replicar um ambiente em produção e assim antecipar alguma anomalia, prever situações de *stress* ou mesmo medir a performance da solução implementada para ajudar a definir e estabelecer *SLA*. Aliado aos testes que são possíveis de realizar, o *Rally* também permite automatizar a instalação de diferentes configurações de serviços do *OpenStack* para que, sobre eles, sejam recolhidos os dados necessários.

De um modo alto nível podemos ver as ações do *Rally* segundo o diagrama que é apresentado na Figura 7.1.

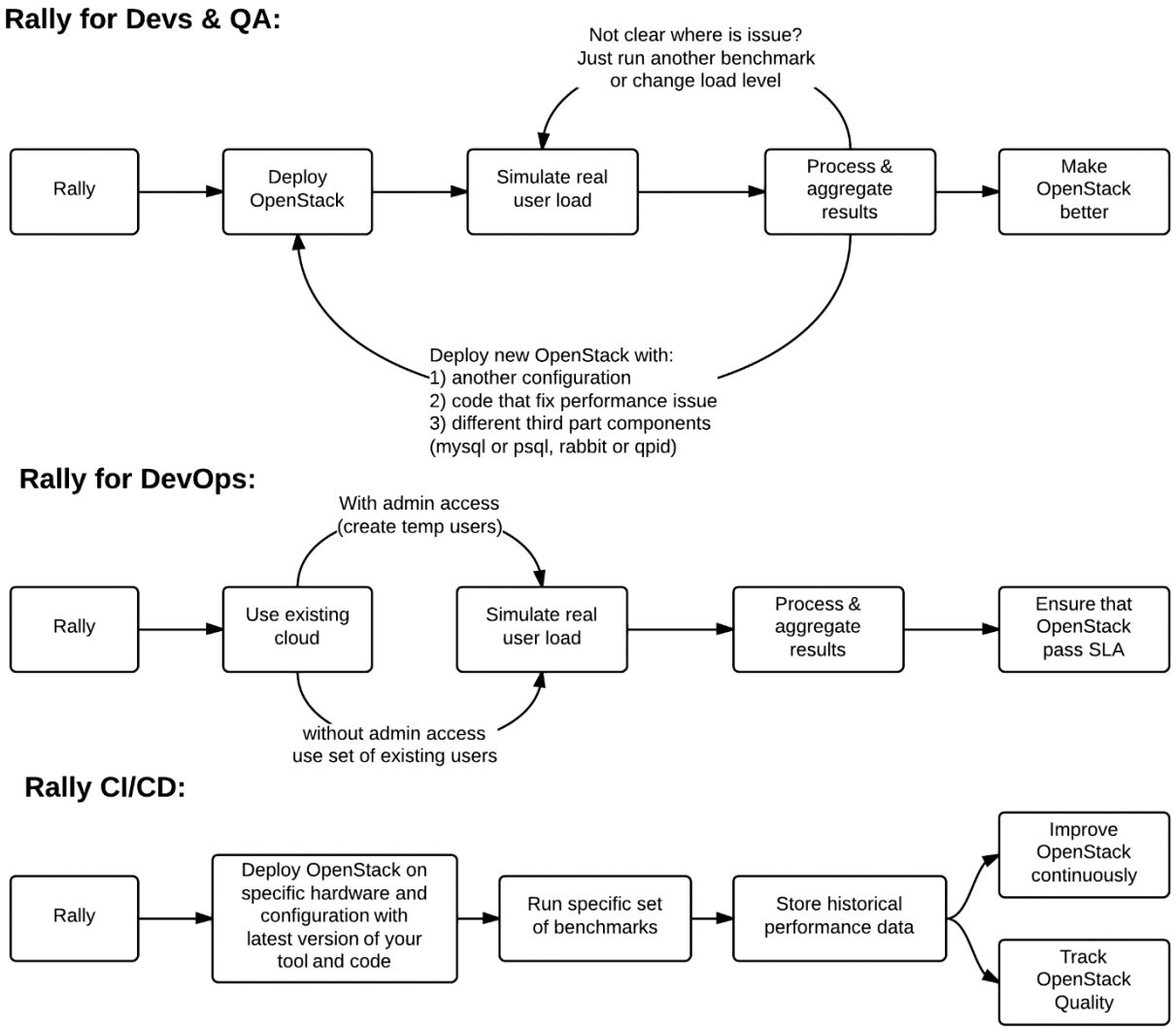

Figura 7.1 - Três principais casos de uso de aplicação do *Rally* (*Rally, 2015*)

Para instalar a aplicação basta clonar o repositório do *GitHub* que é disponibilizado (*Benchmark System for OpenStack, 2015*) e executar o script de instalação que lá se encontra, sendo este um processo bastante simples e de fácil execução. A instalação da aplicação foi feita numa máquina, também ela com sistema operativo Linux, em separado da plataforma de *Cloud*.

Depois da instalação concluída foi necessário registar a plataforma, ou seja, indicar qual a solução de *Cloud* onde deverão ser executados os testes. Este registo passa por identificar o caminho para as *API* públicas do *OpenStack* e fornecer credenciais válidas para login. Isto permite que o *Rally* se autentique como se de um utilizador se tratasse e tenha a possibilidade de automatizar os testes.

Neste momento deverá ser possível, dentro do *Rally* e com linhas de comando dele, executar listagens dos serviços do *OpenStack* e ver o estado de cada um deles. Estes testes permitem perceber, por exemplo, se as credenciais são válidas ou se os endereços para as *API* estão bem configurados de modo a se conseguir comunicar e recolher os dados para cada um dos testes.

O *Rally* permite realizar um conjunto enorme de ações, desde a definição de cenários para implantação de diferentes configurações do *OpenStack*, criação de *plugins* que agregam vários cenários e tarefas para a automatização de testes para os vários serviços da solução de *Cloud*.

No repositório do *Github*, juntamente com os ficheiros de instalação do *Rally*, está disponível uma biblioteca de ficheiros do tipo *json* e *yaml* com os vários testes que podem ser utilizados e/ou reconfigurados de acordo com o pretendido.

A Tabela 7.1 apresenta a estrutura genérica de uma tarefa do *Rally*. As especificações de cada um dos serviços que se pretendam testar vão ditar os componentes e os parâmetros a utilizar em cada situação.

Tabela 7.1 - Exemplo da estrutura de uma tarefa do *Rally*

`ScenarioClass.scenario_method:`
`...`
`args:`
`...`
`runner:`
`...`
`context`
`...`
`sla:`
	Nome completo do cenário de *benchmark*
	
	Args: Utilizado para passagem de parâmetros para a função de *benchmark*. Por exemplo, qual o *flavor* e imagem usar para inicializar as VMs.
	Runner: Especifica o tipo de carga que será gerada no teste. Por exemplo, quantas vezes ou com que frequência de tempo executar o teste.
	Context: Define o ambiente da plataforma. Por exemplo quantos projetos e quantos utilizadores por projeto deve simular.
	SLA: Define o critério de sucesso do teste. Por exemplo, só considera como sucesso caso a percentagem de falhas ou erros seja inferior a uma determinada percentagem.

A Figura 7.2 apresenta o exemplo de uma das tarefas utilizadas para a realização dos testes apresentados neste capítulo. Os vários testes realizados recorreram a esta mesma tarefa mas com diferentes parametrizações. O que esta tarefa vai fazer é executar o teste de *boot* seguido da eliminação das máquinas virtuais criadas. De seguida é escolhido o flavor – nome dado ao elemento que internamente no OpenStack define características como memória, CPU e disco a utilizar na criação de uma instância – e a imagem a utilizar, neste exemplo, o *Ubuntu*. O teste vai ser executados dez vezes e contempla o lançamento de duas instâncias em simultâneo. O ambiente simula a existência de três projetos com dois utilizadores cada.

```
{
    "NovaServers.boot_and_delete_server": [
        {
            "args": {
                "flavor": {
                    "name": "cfg1.mem4"
                },
                "image": {
                    "name": "Openstack Ubuntu 12.0.4 x64"
                },
                "force_delete": false
            },
            "runner": {
                "type": "constant",
                "times": 10,
                "concurrency": 2
            },
            "context": {
                "users": {
                    "tenants": 3,
                    "users_per_tenant": 2
                }
            }
        }
    ]
}
```

Figura 7.2 - Tarefa do *Rally* para teste do *Nova*

Para os vários testes realizados no âmbito deste trabalho prático foram utilizados quatro imagens diferentes, o *Ubuntu* e três versões do *Windows*, o 7, 8 e o *Server* 2012. Foram ainda utilizadas várias combinações de concorrência, e *flavors* de acordo com as condições definidas para estes testes a indicadas mais abaixo na Tabela 7.4.

Após a execução deste teste o *Rally* apresenta a informação de duas maneiras. Uma delas é na linha de comandos logo após a execução do pedido, a outra é graficamente. A informação relativa a cada teste realizado é guardada numa base de dados e posteriormente há a possibilidade de exportar esses dados para um ficheiro *HTML* de modo a serem analisados num qualquer *browser* de Internet conforme pode ser visto na Figura 7.3, o exemplo disso mesmo.

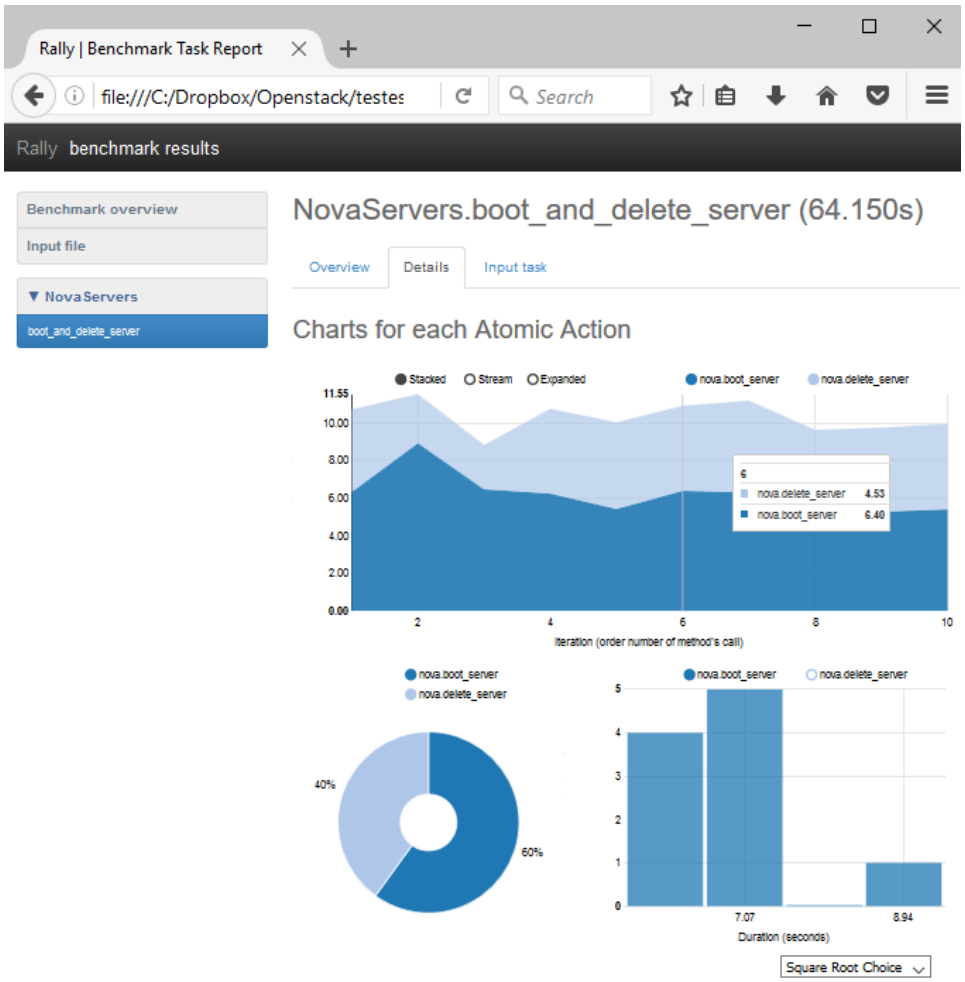

Figura 7.3 - Apresentação gráfica de um teste com o *Rally*

Na lateral esquerda da página são apresentados todos os cenários que compõem o teste executado, que neste caso foi o *boot* e *delete* de uma instância. Na parte mais à direita é apresentado o *Overview* do teste, com o gráfico semelhante ao da Figura 7.3 com o total do tempo para cada iteração, e uma tabela com valores máximos, mínimos e média. O separador *Details* que é aqui apresentado mostra um gráfico detalhado por cada uma das ações tomadas no teste, já o separador *Input task* mostra o código do teste realizado, igual ao código visto anteriormente na Figura 7.2.

7.3. Setup Experimental

Os testes realizados pretendem avaliar a performance da plataforma medindo o tempo de resposta aos pedidos de criação e eliminação de máquinas virtuais. Adicionalmente pretende-se analisar quais os parâmetros de configuração das máquinas virtuais influenciam esses tempos. Para tal foram realizados um conjunto de testes que pretendem abranger vários cenários possíveis.

Terminologia utilizada doravante na explicação da análise realizada:

- **Teste**: Pedido de criação e eliminação de máquinas virtuais com recolha individual do tempo de cada ação. Cada teste foi realizado mil vezes, tendo sido considerada a média aritmética de cada ação.

- **Concorrência**: Número de máquinas virtuais pedidas em simultâneo num teste. Vão ser realizados testes de concorrência 2, 4 e 6.

No que concerne aos sistemas operativos a utilizar nas máquinas virtuais, decidiu-se, por uma questão de diversidade na oferta face às necessidades previstas e para espelhar a utilização interna, disponibilizar desde logo uma versão *Linux* e três versões do *Windows*. O detalhe de cada um dos sistemas operativos utilizados encontra-se descrito na Tabela 7.2. De salientar que o utilizador da plataforma tem a possibilidade de adicionar outras imagens ao seu projeto.

Com exceção da imagem do *Windows Server 2012 R2*, que foi recolhida do site *Cloudbase Solutions* (*Cloudbase, 2014*), todas as restantes imagens foram criadas e configuradas de raiz no âmbito deste projeto. Não foi utilizada mais nenhuma "*Estação Padrão*" em produção interna como aconteceu no teste realizado em ambiente real, uma vez que as máquinas virtuais foram apenas criadas com vista à realização de testes aplicacionais e provas de conceito, simulando ambientes não padronizados ao contrário daqueles que existem internamente.

Tabela 7.2 - Detalhe dos sistemas operativos disponibilizados nas imagens da plataforma

Sistema Operativo	Versão
Ubuntu	Desktop 12.0.4, 64 bits
Windows 7	Professional SP1, 64 bits
Windows 8	Professional, 64 bits
Windows Server 2012	R2 Standard Evaluation for Openstack, 64 bits

Cada teste tem uma conjugação de parâmetros de sistema operativo, memoria, disco e CPU com valores diferentes. O parâmetro memória vai assumir configurações com valores de 1024Mb, 2048Mb e 4096Mb. O disco rígido vai ter valores de 30Gb, 60Gb e 120Gb, por sua vez o parâmetro CPU vai ser configurado para 2 *cores*, 4 *cores* e 8 *cores*. Em virtude do enorme número de conjugações possíveis entre os vários parâmetros em estudo, decidiu-se que, quando é analisado um dos parâmetros, memória, disco ou CPU os restantes assumem os valores base que se encontram representados na Tabela 7.3.

Por exemplo, quando são testados as três configurações de memória, o tamanho do disco e o número de cores do *CPU* de cada instância é fixado nesses valores, ou seja, 60Gb e 4 núcleos respetivamente.

Os valores de base foram escolhidos segundo as configurações típicas que uma máquina virtual irá assumir quando for configurada na plataforma.

Tabela 7.3 - Valores base para os parâmetros de memória, disco e *CPU*

Memória RAM (Mb)	Disco Rígido (Gb)	CPU (número de cores)
4096	60	4

Cada conjugação dos componentes, sistema operativo, memória, disco e *CPU* foi ainda testada segundo três níveis de concorrência, ou seja, os testes representados na Tabela 7.4 foram realizados três vezes, a primeira para medir o tempos de pedidos com duas máquinas, a segunda para pedidos com 4 máquinas e a última para pedidos de 6 máquinas em simultâneo.

Cada teste realizado contempla uma conjugação diferente de parâmetros, conjugação essa que se encontra resumida na Tabela 7.4, em que, cada linha apresenta a configuração das máquinas virtuais desse teste.

Cada valor de tempo "*t(s)*" é medido em segundos e corresponde ao tempo de boot e de delete registado num teste. Foi definida esta escala temporal por ser a que melhor se adaptava ao que se pretendia medir e levando em consideração o *hardware* utilizado. O valor apresentado para cada teste corresponde à média aritmética de 1000 iterações.

Tabela 7.4 - Conjugação de parâmetros e variação dos mesmos para a recolha de informação

				Concorrência	
S.O.	**Memória**	**Disco**	**CPU**	**Tempo Boot**	**Tempo Delete**
Ubuntu	1024	60	4	t(s)	t(s)
Win7	1024	60	4	t(s)	t(s)
Win8	1024	60	4	t(s)	t(s)
Ws2012	1024	60	4	t(s)	t(s)
Ubuntu	2048	60	4	t(s)	t(s)
Win7	2048	60	4	t(s)	t(s)
Win8	2048	60	4	t(s)	t(s)
Ws2012	2048	60	4	t(s)	t(s)
Ubuntu	4096	60	4	t(s)	t(s)
Win7	4096	60	4	t(s)	t(s)
Win8	4096	60	4	t(s)	t(s)
Ws2012	4096	60	4	t(s)	t(s)
Ubuntu	4096	30	4	t(s)	t(s)
Win7	4096	30	4	t(s)	t(s)
Win8	4096	30	4	t(s)	t(s)
Ws2012	4096	30	4	t(s)	t(s)
Ubuntu	4096	60	4	t(s)	t(s)
Win7	4096	60	4	t(s)	t(s)
Win8	4096	60	4	t(s)	t(s)
Ws2012	4096	60	4	t(s)	t(s)
Ubuntu	4096	120	4	t(s)	t(s)
Win7	4096	120	4	t(s)	t(s)
Win8	4096	120	4	t(s)	t(s)
Ws2012	4096	120	4	t(s)	t(s)
Ubuntu	4096	60	2	t(s)	t(s)
Win7	4096	60	2	t(s)	t(s)
Win8	4096	60	2	t(s)	t(s)
Ws2012	4096	60	2	t(s)	t(s)
Ubuntu	4096	60	4	t(s)	t(s)
Win7	4096	60	4	t(s)	t(s)
Win8	4096	60	4	t(s)	t(s)
Ws2012	4096	60	4	t(s)	t(s)
Ubuntu	4096	60	8	t(s)	t(s)
Win7	4096	60	8	t(s)	t(s)
Win8	4096	60	8	t(s)	t(s)
Ws2012	4096	60	8	t(s)	t(s)

7.4. Avaliação do OpenStack

Os testes realizados com recurso à plataforma *Rally* permitiram-nos recolher informação sobre a performance da plataforma em que foi medido o tempo de resposta dos pedidos de criação e eliminação de máquinas virtuais. Estes testes contemplaram pedidos em simultâneo de duas, quatro ou seis máquinas virtuais com diferentes configurações de memória, disco e CPU. A análise dos dados é apresentada de seguida.

7.4.1. Variação de memória

A análise feita neste ponto considera a flutuação de memória para 1Gb, 2 Gb e 4 Gb por cada máquina virtual, tendo sido fixados os valores do disco em 60Gb e do CPU em 4 *cores*, valores tidos como base para este estudo.

A Tabela 7.5 mostra que para pedidos de duas máquinas em simultâneo, e com isto se depreende pouco esforço solicitado à plataforma, o tempo despendido no arranque das máquinas pouco difere com a variação dos parâmetros da memória, registando-se uma diferença de 1,45s entre o pedido que demorou menos tempo e o pedido que demorou mais, verificado com máquinas de 1Gb de memória e 4Gb respetivamente.

Tabela 7.5 - Valores mínimos e máximos para cada tipo de concorrência (memória)

Concorrência	Min	Max	Diferença
Conc. 2	5,52 (1Gb)	6,97 (4Gb)	1,45s
Conc. 4	8,41 (2Gb)	10,46 (2Gb)	2,05s
Conc. 6	11,36 (2Gb)	13,73 (4Gb)	2,37s

Analisando o gráfico da Figura 7.4 consegue extrapolar-se uma tendência de subida nos tempos de *boot* quando se aumenta a memória das *VM's* de 1Gb para 2Gb e para 4Gb, com maior incidência quando a concorrência é de 6 instâncias. Os resultados mostram que o aumento da concorrência faz aumentar o tempo de resposta.

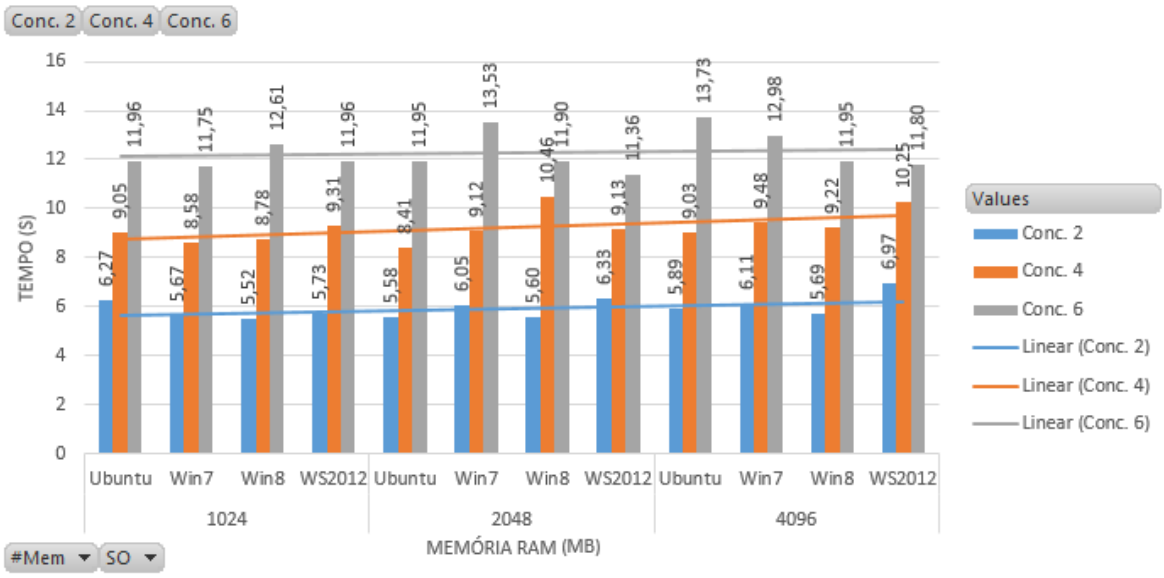

Figura 7.4 - Tempo de *boot* por variação de *SO* para cada memória e concorrência

Na visualização apresentada pela Figura 7.5 foram tidos como base os mesmos valores usados na construção do gráfico anterior, no entanto, os resultados são apresentados com a vista por sistema operativo, ao que se confirma a mesma linha de tendência no tempo de *boot* que é

proporcional à concorrência (o aumento da concorrência faz aumentar o tempo e *boot*). De realçar que o *Windows 8* tem um melhor comportamento com concorrência 2, onde apresenta os valores mais baixos em qualquer uma das configurações de memória. Com piores prestações aparece o *Windows Server 2012* com os tempos mais altos para o *boot* de 4 instâncias em simultâneo e o *Windows 7* para o *boot* de 6.

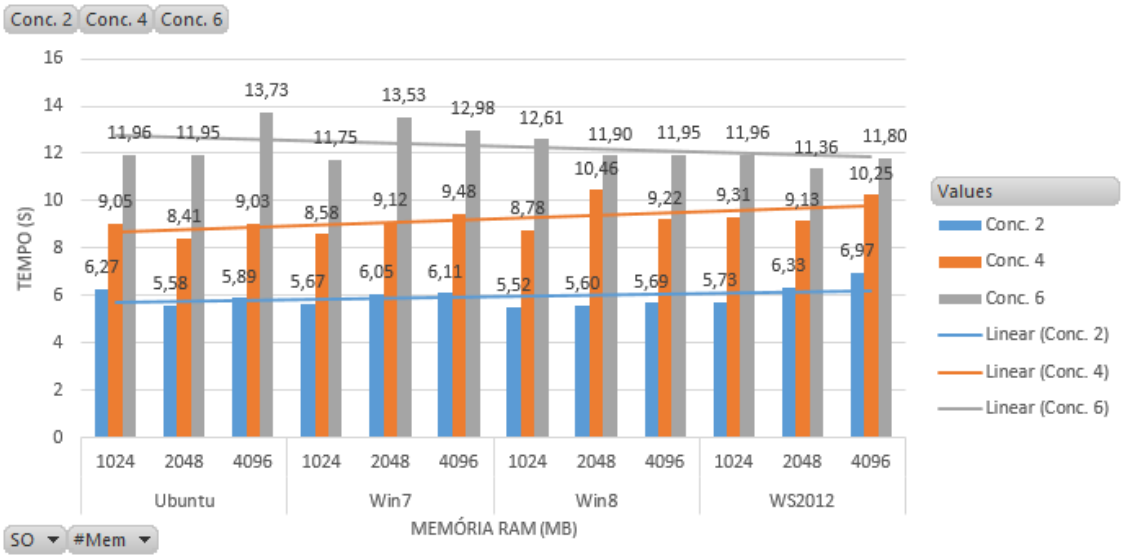

Figura 7.5 - Tempo de *boot* por variação de memória para cada *SO* e concorrência

O gráfico apresentado na Figura 7.6 não contém a variável sistema operativo, apresentando apenas média dos seus tempos para cada configuração de memória e cada concorrência.

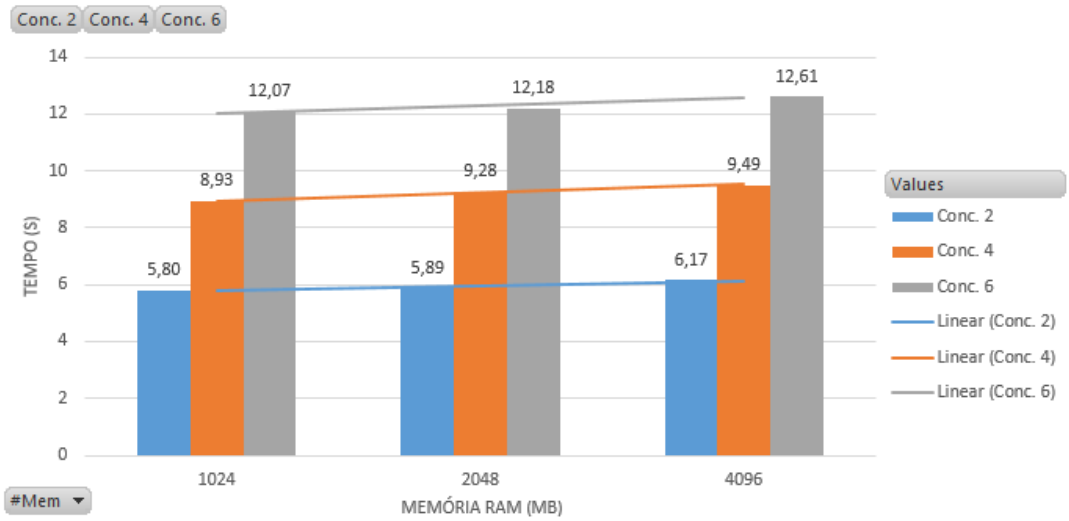

Figura 7.6 - Tempo de *boot* para variação de memória

Conforme esperado, confirma-se a proporcionalidade entre o tempo de *boot* e o aumento do número de máquinas por pedido. A Tabela 7.6 mostra que para qualquer uma das configurações de memória o tempo de *boot* mais do que duplica quando se passa de 2 máquinas em simultâneo para um pedido de 6 máquinas.

Tabela 7.6 - Percentagem de aumento do tempo de *boot* com o aumento da concorrência (memória)

	Conc. 2 para Conc. 4	Conc. 4 para Conc. 6	**Conc. 2 para Conc. 6**
1024 Mb	54,0%	35,1%	**108,1%**
2048 Mb	57,6%	31,3%	**106,9%**
4096 Mb	54,0%	32,8%	**104,5%**

Com a Tabela 7.6 extrapola-se uma tendência que é validada com a Tabela 7.7 em que, para cada patamar de concorrência o aumento do tempo de *boot* não é crescente.

Tabela 7.7 - Percentagem de aumento do tempo de *boot* com o aumento da memória

	1024Mb para 2048Mb	2048Mb para 4096Mb	**1024Mb para 4096Mb**
Conc. 2	1,6%	4,7%	**6,4%**
Conc. 4	4,0%	2,3%	**6,3%**
Conc. 6	1,0%	3,5%	**4,5%**

Como complemento ao estudo apresentado foi feita uma amostragem dos testes realizados, que consistiu na recolha dos tempos de 10 testes.

Consegue-se perceber claramente pela Figura 7.7 que existe uma correlação entre a concorrência e o tempo de *boot*, o mesmo acontece para a variação de memória embora não de uma maneira tão expressiva.

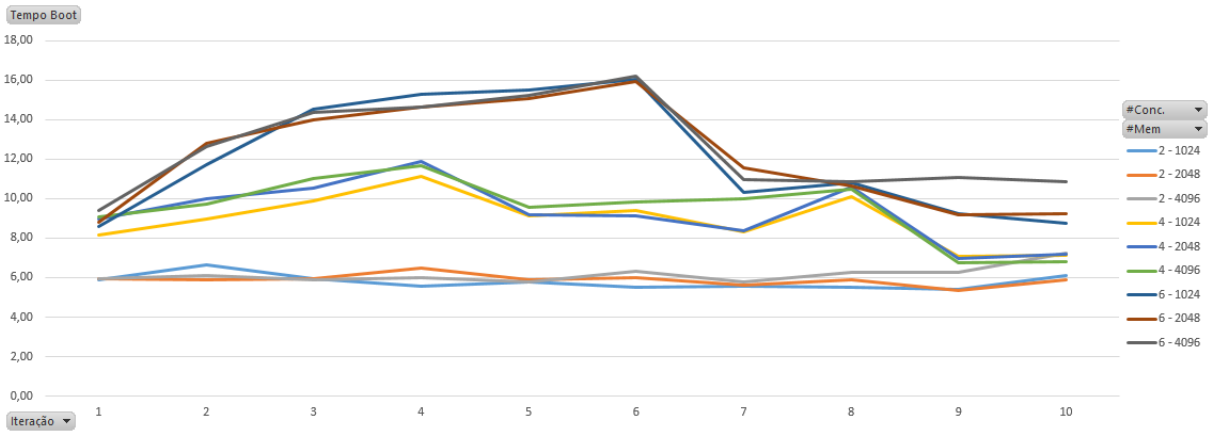

Figura 7.7 - Tempo de *boot* em cada teste numa amostragem de 10 pedidos (Memória)

Outra avaliação adicional realizada com a mesma amostra foi a medição do tempo que a plataforma demorou a fazer o *delete* das máquinas virtuais após a sua criação. Perante a análise da Figura 7.8 pode-se dizer que não se verifica uma relação marcada entre os componentes de memória e concorrência face à performance da plataforma, contudo, o destaque de tempos que aparece no gráfico refere-se a pedidos de 6 *VM* em simultâneo. Depreende-se no entanto, que o tempo de eliminação de um determinado conjunto de máquinas é no máximo 8 segundos, que é metade do tempo máximo utilizado para a execução de um pedido de 6 instâncias.

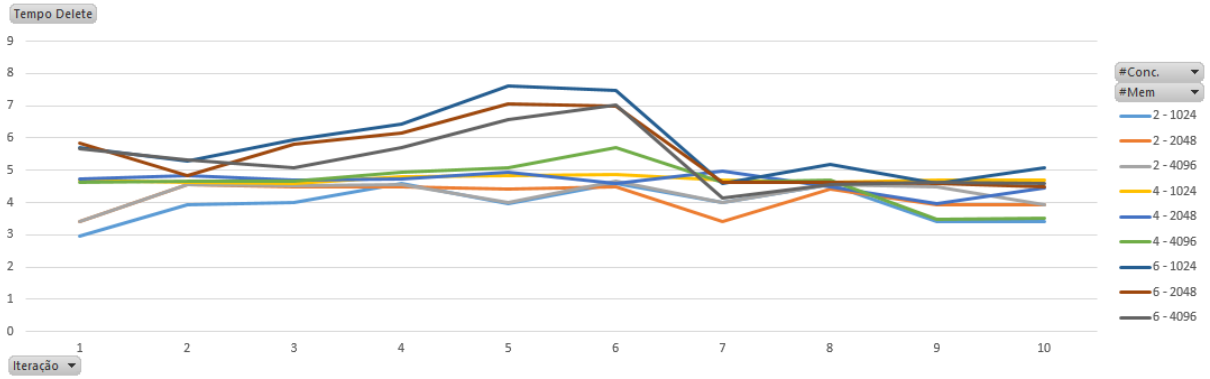

Figura 7.8 - Tempo de *delete* em cada teste numa amostragem de 10 pedidos (Memória)

7.4.2. Variação de disco

A análise feita neste ponto considera a flutuação da capacidade de disco rígido alocado a cada instância para valores de 30 Gb, 60 Gb e 120 Gb, fixado os valores da memória *RAM* em 4Gb e do *CPU* em 4 *cores*, valores tidos como base para este estudo.

A variação nos tempos de arranque ao nível da variação do disco não é tão expressiva ou notória como no caso da memória. Não é possível verificar uma tendência de crescimento para além da esperada, causada pelo aumento da concorrência. A razão para esta situação poderá estar relacionada com o tipo de alocação que é feito ao nível do espaço do disco rígido, o que não acontece com a memória e com o número de núcleos de *CPU*. O disco não é alocado na sua totalidade, ou seja, o espaço total não é reservado à partida. Ao invés é reservado o espaço associado ao tamanho da imagem e o espaço necessário para esta se expandir e dar lugar a ficheiros do sistema operativo. O valor especificado para o disco é informativo no que refere até que tamanho pode expandir-se e não do tamanho inicial efetivo, resultando nos valores apresentados tão nivelados entre si conforme se verifica na Tabela 7.8 e mais detalhadamente na Figura 7.9.

Tabela 7.8 - Valores mínimos e máximos para cada tipo de concorrência (disco rígido)

Concorrência	Min	Max	Diferença
Conc. 2	5,49 (30Gb)	6,83 (120Gb)	1,34s
Conc. 4	8,85 (30Gb)	9,99 (120Gb)	1,14s
Conc. 6	11,94 (60Gb)	13,07 (60Gb)	1,13s

Neste caso a maior diferença é de 1,34s, que é inferior à mais pequena nos testes de memória, que se situa nos 1,45s.

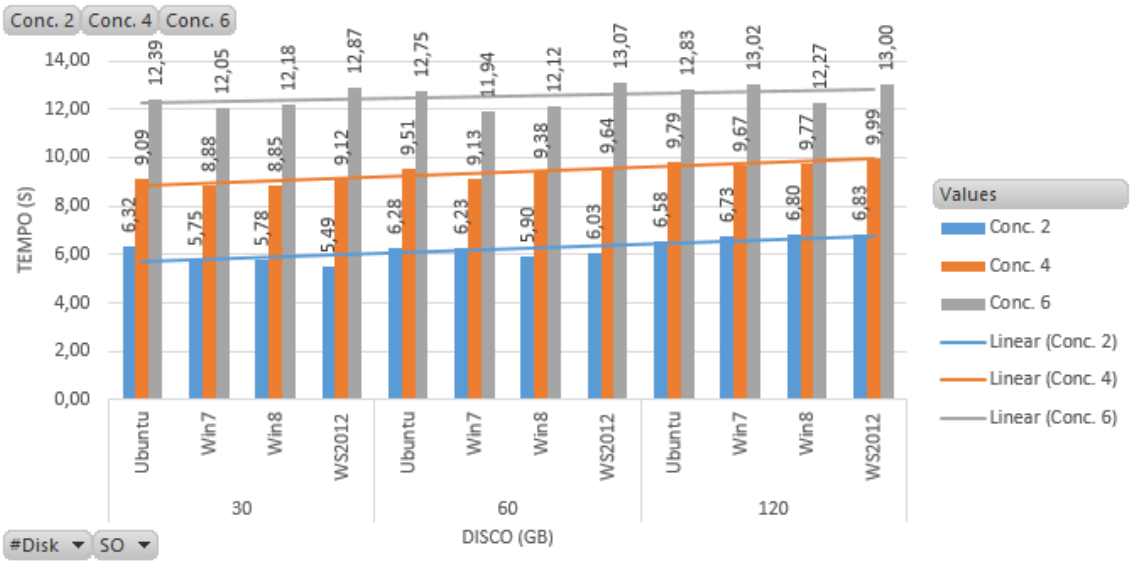

Figura 7.9 - Tempo de *boot* por variação de *SO* para cada tamanho de disco rígido e concorrência

Pela análise do gráfico da Figura 7.10 foram tidos como base os mesmos valores usados na construção do gráfico anterior, no entanto os resultados são apresentados com a vista por sistema operativo. Percebe-se assim que, para cada um, existe um aumento no tempo de *boot* entre 0,5 e 1 segundo quando se aumenta o tamanho do disco associado a cada instância.

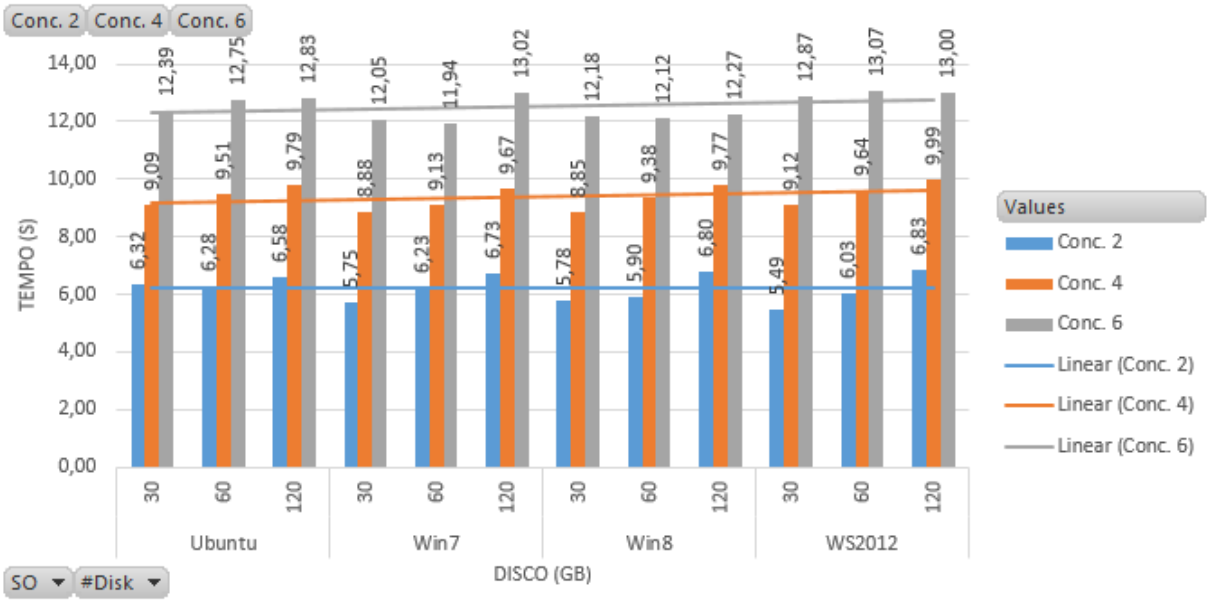

Figura 7.10 - Tempo de *boot* por variação do tamanho do disco para cada *SO* e concorrência

O gráfico apresentado na Figura 7.11 não contém a variável sistema operativo, apresentando apenas média dos seus tempos para cada configuração de memória e cada concorrência.

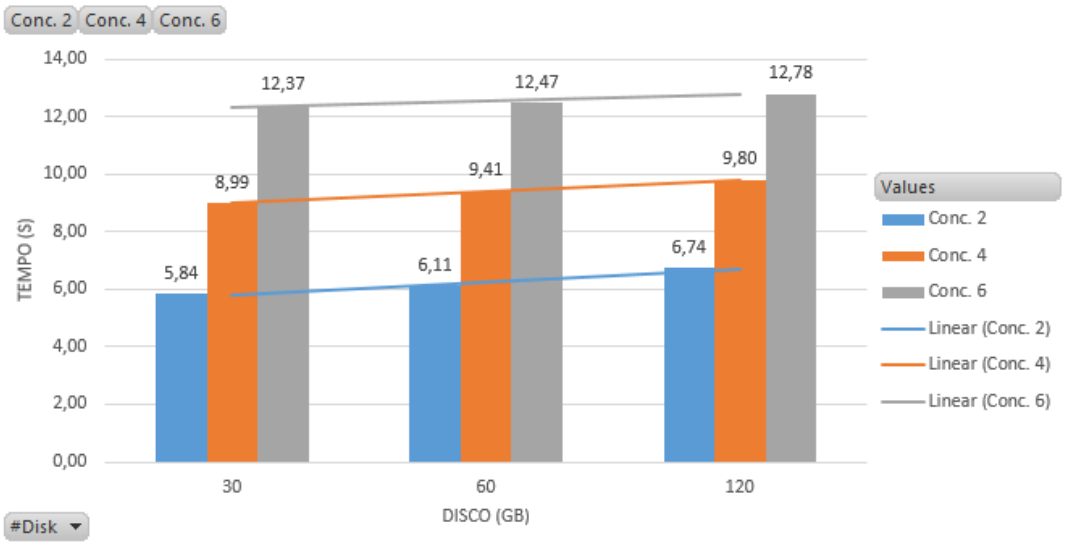

Figura 7.11 - Tempo de *boot* para variação de disco

Como mostra a Figura 7.11 e pelas percentagens da Tabela 7.9, para cada uma das configurações do disco existe de facto um aumento do tempo de *boot* das máquinas com o aumento da concorrência.

Tabela 7.9 - Percentagem de aumento do tempo de *boot* com o aumento da concorrência (disco)

	Conc. 2 para Conc. 4	Conc. 4 para Conc. 6	**Conc. 2 para Conc. 6**
30 Gb	54,05	37,7%	**112,0%**
60 Gb	54,1%	32,4%	**104,1%**
120 Gb	45,6%	30,4%	**89,8%**

Segundo a Tabela 7.10, ao analisar cada patamar de concorrência e o salto dos 30Gb para os 60Gb e o seguinte, dos 60Gb para os 120Gb, há de facto um aumento percentual do tempo de *boot*.

Tabela 7.10 - Percentagem de aumento do tempo de *boot* com o aumento do disco

	30Gb para 60 Gb	60 Gb para 120 Gb	**30 Gb para 120 Gb**
Conc. 2	4,7%	10,2%	**15,4%**
Conc. 4	4,8%	4,1%	**9,1%**
Conc. 6	0,8%	2,5%	**3,3%**

Os dados da experimentação induzem uma correlação entre a concorrência e o tempo de *boot*, tal como aconteceu no caso da variação de memória. A diferença para a análise anterior está na separação mais notória que existe entre os tempos de *boot* para variação do disco em cada um dos patamares de concorrência. Analisando os tempos individuais de uma amostra denota-se uma correlação mais forte entre o tamanho do disco e a performance.

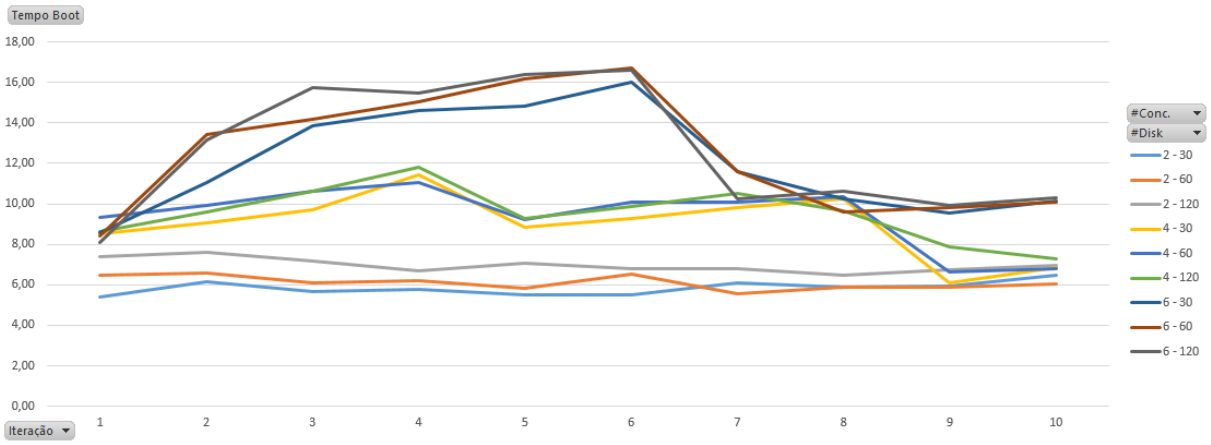

Figura 7.12 - Tempo de *boot* em cada teste numa amostragem de 10 pedidos (Disco)

Outra avaliação adicional realizada com a mesma amostra foi a medição do tempo que a plataforma leva a fazer o *delete* das máquinas virtuais após a sua criação. A análise do gráfico da Figura 7.13 para os tempos de *delete* é em tudo igual à realizada no caso anterior. Para este caso, há também alguns valores que se destacam para os pedidos de 6 máquinas em simultâneo, no entanto, nada de relevante face ao cenário global da amostra. Para este caso mediu-se menos 1s para o tempo máximo de *delete* das máquinas virtuais.

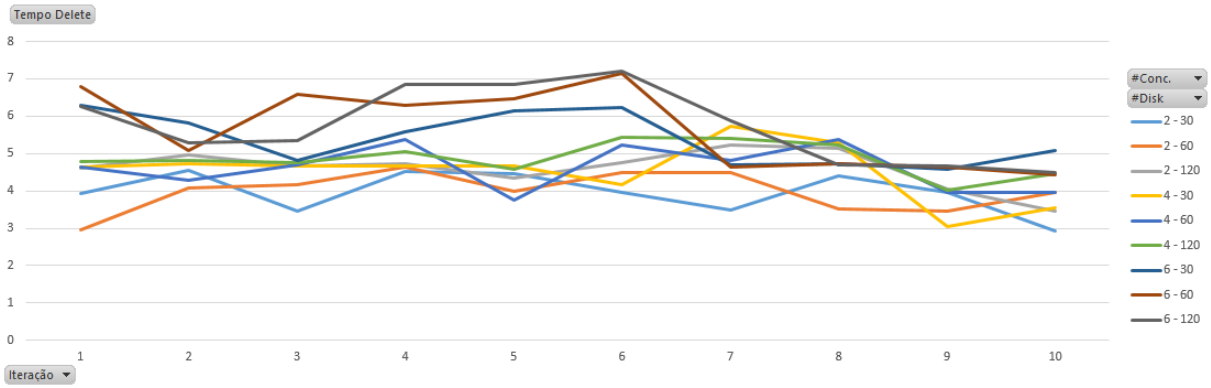

Figura 7.13 - Tempo de delete em cada teste numa amostragem de 10 pedidos (Disco)

7.4.3. Variação de CPU

A análise feita neste ponto considera a flutuação da quantidade de núcleos por *CPU* para quantidade de 2, 4 e 6 *cores* para cada máquina virtual, tendo sido fixados os valores de memória em 4Gb e do disco rígido em 60 Gb, valores tidos como base para o este estudo.

Para pedidos de duas máquinas em simultâneo, e com isto se depreende pouco esforço solicitado à plataforma, o tempo despendido no arranque das máquinas pouco difere com a variação do número de núcleos de *CPU*. Pela análise da Tabela 7.11, são 0,66s de diferença entre o valor mais baixo de 5,7s para 2 *cores* e os 6,36s para os 8 *cores*, valor este que é menos de metade da diferença encontrada para as variações da memória e disco que foram de 1,34s e 1,45s respetivamente. Extrapola-se uma tendência de aumento constante, cerca de 1s, entre os valores mínimos e máximos de *boot* quando se aumenta a concorrência para 4 e para 6 máquinas virtuais.

Tabela 7.11 - Valores mínimos e máximos para cada tipo de concorrência (CPU)

Concorrência	Min	Max	Diferença
Conc. 2	5,70 (2Cores)	6,36 (8 Cores)	0,66s
Conc. 4	8,26 (8Cores)	9,89 (8 Cores)	1,63s
Conc. 6	11,20 (2Cores)	13,89 (8Cores)	2,69s

Com recurso à Figura 7.14 percebe-se que para pedido com concorrência 2 o aumento do *CPU* pouco ou nada influencia o tempo de *boot*, como pode ver-se no gráfico a azul. No entanto nota-se uma tendência de subida no sentido do aumento dos *CPU* e da concorrência. Esse aumento é mais notório para a concorrência 6 e na passagem de 4 para os 8 *cores* por instância.

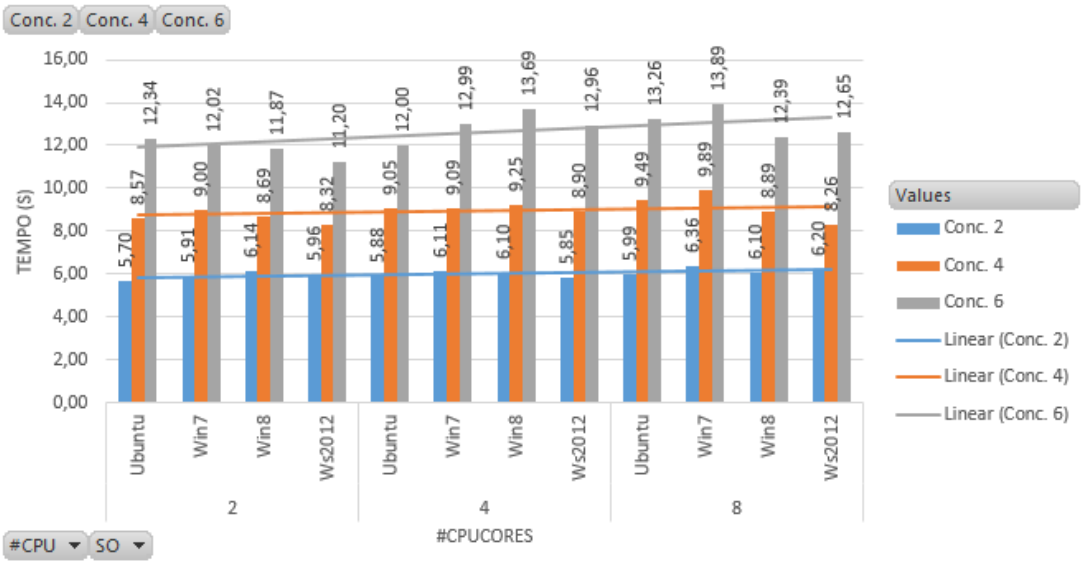

Figura 7.14 - Tempo de *boot* por variação de *SO* para cada memória e tipo de concorrência

Na visualização dos dados apresentada na Figura 7.15 foram tidos como base os mesmos valores usados na construção do gráfico anterior, no entanto, os resultados apresentados são focados no sistema operativo e confirma-se a mesma linha de tendência no tempo de *boot* que aumenta com o aumento da concorrência. Adicionalmente é visto que o *Windows Server 2012* tem um melhor comportamento para pedidos com concorrência 4 e 6, onde apresenta os valores mais baixos, independente do numero de núcleos, face aos outros sistemas operativos.

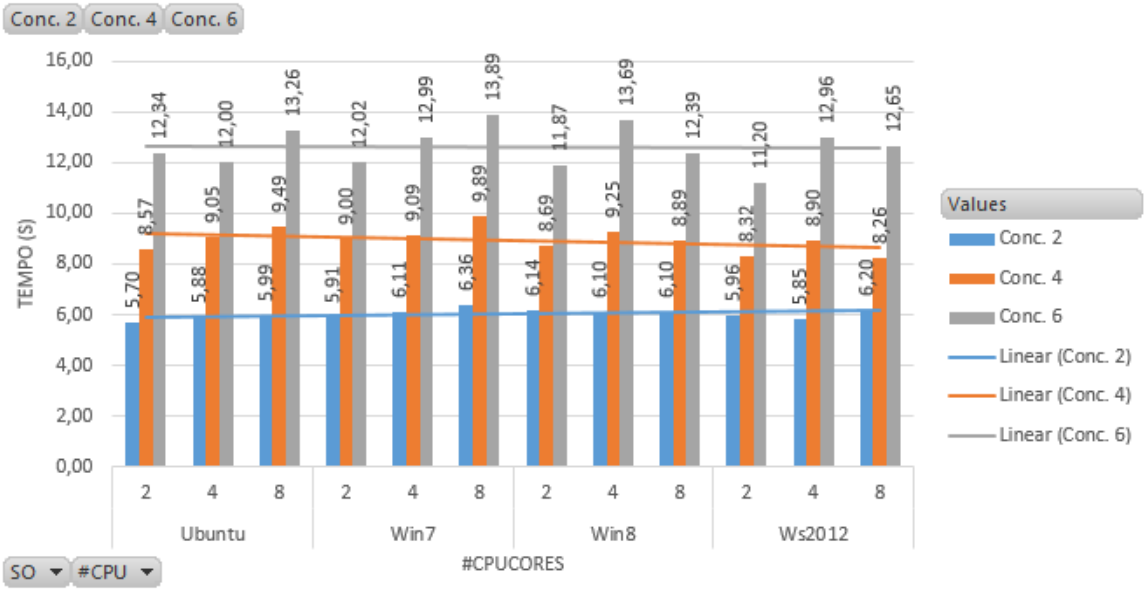

Figura 7.15 - Tempo de *boot* por variação de *CPU* para cada *SO* e tipo de concorrência

Para o gráfico apresentado na Figura 7.16 foi retirada a variável sistema operativo, apresentando apenas a média dos seus tempos para cada configuração do número de núcleos atribuído a cada máquina virtual.

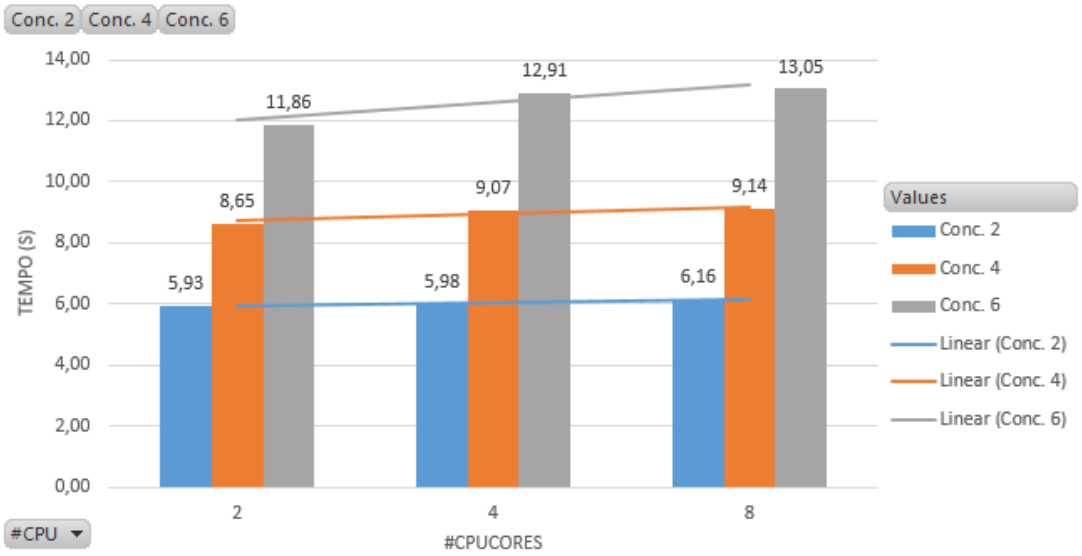

Figura 7.16 - Tempo de *boot* para variação de *CPU*

Conforme esperado, confirma-se uma subida do tempo de *boot* com o aumento do número de máquinas por pedido. A Tabela 7.12 mostra que para qualquer uma das configurações de *CPU* o tempo de *boot* é bastante mais elevado, especialmente quando se passa de 2 máquinas em simultâneo para um pedido de 6 máquinas, aumento superior face ao registado com a oscilação do parâmetro memória.

Tabela 7.12 - Percentagem de aumento do tempo de *boot* com o aumento da concorrência (*CPU*)

	Conc. 2 para Conc. 4	Conc. 4 para Conc. 6	**Conc. 2 para Conc. 6**
Cpu2	45,9%	37,2%	100,1%
Cpu4	51,6%	42,2%	115,7%
Cpu8	48,2%	42,8%	111,7%

Na Tabela 7.13 note-se que na análise da flutuação dos valores do *CPU* a percentagem de aumento de tempo verifica-se nos dois sentidos. Quando se analisa a passagem de 2 para 8 núcleos depreende-se uma influência positiva no tempo de *boot* das máquinas virtuais com 4% de incremento do tempo na concorrência 2, aumentando para 5,7% na concorrência 4 e para 10% para pedidos de 6 máquinas em simultâneo, havendo novamente um aumento superior em relação ao da concorrência menor.

Tabela 7.13 - Percentagem de aumento do tempo de *boot* com alteração do *CPU*

	Cpu2 para Cpu4	Cpu4 para Cpu8	**Cpu2 para Cpu8**
Conc. 2	1,0%	3,0%	4,0%
Conc. 4	5,0%	0,7%	5,7%
Conc. 6	8,9%	1,1%	10,0%

A análise do gráfico da Figura 7.17 torna clara a existência de uma correlação entre a concorrência e o tempo de *boot*, tal como aconteceu nos casos analisados anteriormente. Avaliando cada um dos patamares de concorrência verifica-se uma maior separação entre os tempos para a oscilação do parâmetro em estudo, o que revela uma correlação do aumento do *CPU* com a performance da plataforma. Esta separação de valores é mais visível em relação à memória mas de menor valor que no caso do tamanho disco.

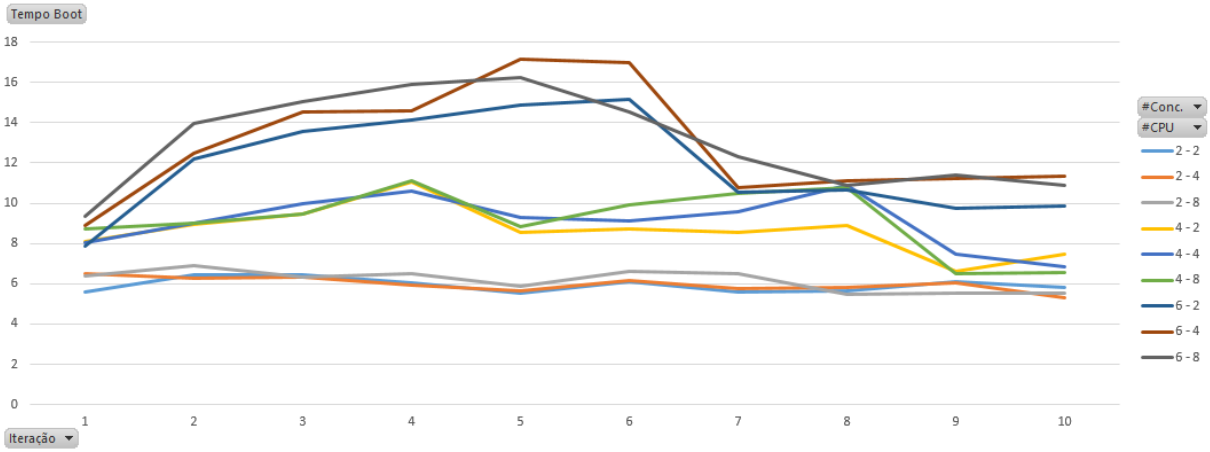

Figura 7.17 - Tempo de *boot* em cada teste numa amostragem de 10 pedidos (*CPU*)

Outra avaliação realizada com a mesma amostra foi a medição do tempo que a plataforma demorou para levar a cabo o *delete* das máquinas virtuais após a sua criação. A análise do gráfico da Figura 7.18 para os tempos de *delete* é em tudo igual à que foi feita para os casos anteriores, no entanto, é de salientar que neste caso se verifica uma maior separação entre os patamares de concorrência.

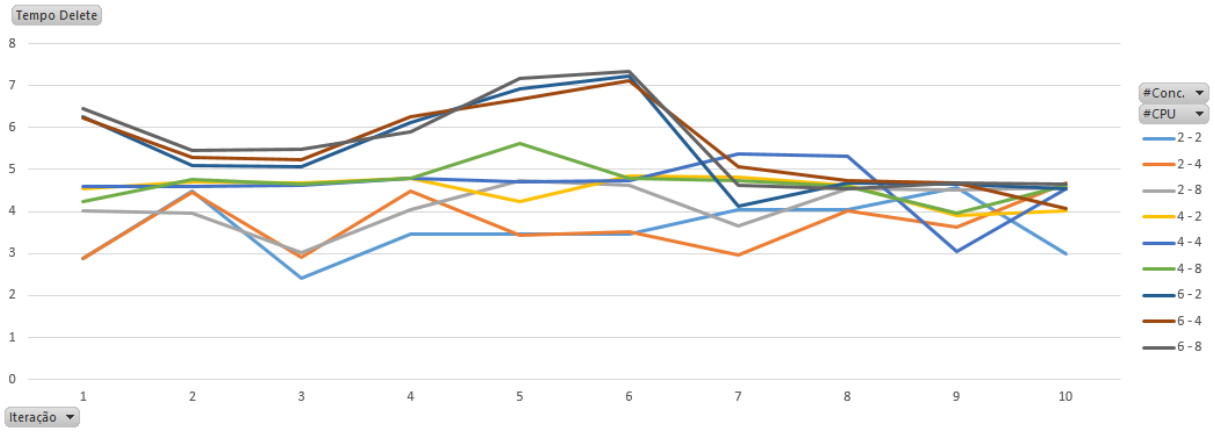

Figura 7.18 - Tempo de delete em cada teste numa amostragem de 10 pedidos (Disco)

7.5. Considerações finais

Nesta análise verificou-se que o tempo de *boot* das máquinas virtuais está diretamente relacionado com o aumento do número de pedidos de execução de máquinas em simultâneo. Em todos os casos de teste, ao triplicar o número de máquinas a executar, o tempo de resposta provou ser duas vezes superior.

Na componente em que se mediu a influência que o aumento das características têm no aumento do tempo de *boot* das máquinas, verificou-se que houve de facto uma alteração mas sempre de valores inferiores a 10%. Embora os valores não sejam expressivos foi possível perceber que a variação do componente disco, seguido do *CPU*, influenciam a performance da plataforma de *Cloud*. Em relação ao tempo de *delete* de máquinas virtuais, os testes realizados foram inconclusivos quanto à sua influência na performance da plataforma.

Estes valores estão dentro dos parâmetros esperados tendo em linha de conta as características do *hardware* utilizado, sendo este de gama comercial e com componentes de características modestas, que está de todo enquadrado com o que era pretendido no levantamento inicial, o de criar uma solução de *Cloud* privada com recursos utilizados diariamente.

8. CONCLUSÕES E TRABALHO FUTURO

As arquiteturas de *Cloud Computing* exploram as técnicas de virtualização para fornecer várias máquinas virtuais no mesmo servidor físico (para o caso de *IaaS*), de modo a utilizar eficientemente os recursos disponíveis. O paradigma de *Cloud Computing* e as soluções disponíveis no mercado que se baseiam no *OpenStack* foram e continuam a ser continuamente exploradas numa grande diversidade de cenários industriais e também académicos. Este tipo de arquitetura tem apresentado desde cedo bons resultados ao nível do desempenho, escalabilidade, flexibilidade, confiabilidade e aceitação final por parte do utilizador.

O OpenStack devido à sua modularidade e leque de oferta variada, com projetos robustos e com novas funcionalidades que trazem mais-valias para as soluções de *Cloud open source*, pode ser considerado uma opção sólida para a implementação de uma solução de *Cloud* privada para as empresas.

A plataforma de *Cloud* não foi criada com vista à execução de tarefas de elevado desempenho – embora o conseguisse fazer com a devida redução no número de máquinas virtuais que ficariam disponíveis – apresenta uma boa relação custo/desempenho. Com esta solução, implementada com recurso a quatro computadores de gama comercial utilizados no dia-a-dia da empresa, consegue-se disponibilizar três vezes mais recursos face ao que os computadores permitiam *per si*, reduzindo custos e controlando recursos de uma forma mais racional.

O trabalho realizado estabeleceu uma base sólida de conhecimento que vai permitir que a solução implementada possa crescer de forma sustentada e continuar a ser considerada como um valor acrescentado para os seus utilizadores.

Com a pretensão de realizar um *upgrade* aos servidores, já se encontra desenhada uma arquitetura mais refinada com a integração do serviço de rede e telemetria, aliados aos sistemas já em produção atualmente, que visa um crescimento mais sustentado e por mais tempo. Ao preparar a solução para crescer torna-se importante não só fazer o acompanhamento, ao monitorizar a infraestrutura, mas também analisar os dados recolhidos com ferramentas analíticas, para extrair conhecimento e otimizar tanto o *hardware* como os serviços implementados.

Como trabalho futuro espera-se que seja possível a atribuição de servidores mais robustos para dessa forma implementar a solução mais recente do *OpenStack* de modo a migrar a plataforma existente e assim integrar com os sistemas de monitorização.

Do ponto de vista pessoal este estágio contribuiu para o meu crescimento pessoal mas acima de tudo como enriquecimento técnico. Considero que foi uma experiencia muito satisfatória de conseguir deixar informação válida e que pode ser reaproveitada no futuro.

REFERÊNCIAS BIBLIOGRÁFICAS

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M. (2009). *Above the Clouds: A Berkeley View of Cloud Computing*. Disponível em http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

Baptista, D., Moreira, E., Quartin, F., Carvalho, J., Neves, J., Carvalho, J., Pimentão, J., Cunha, L., Santos, L., Vidigal, L., Santos, M., Carvalho, P., Vilela, P., Duque, P., Cristo, R., Abreu, S., Luís, S. (2004). *Open Source Software – Que oportunidades em Portugal?*. APDSI – Associação para a Promoção e Desenvolvimento da Sociedade de Informação.

Barkat, A., Santos, A., Ho, T. 2014. *OpenStack and CloudStack: Open Source Solutions for Building Public and Private Clouds*. 2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. DOI 10.1109/SYNASC.2014.64

Benchmark System for OpenStack. Consultado em Outubro, 2015, em https://github.com/openstack/rally

Bist, M., Wariya, M., Agarwal, A. 2013. *Comparing Delta, Open Stack and Xen Cloud Platforms: A Survey on Open Source IaaS*. 2013 3rd IEEE International Advance Computing Conference (IACC)

Borenstein, N., Blake, J. (2011). *Cloud Computing Standards: Where's the Beef?*. IEEE Internet Computing, Vol. 15, No. 3, pp. 74-78

Borhani, A., Leitner†, P., Lee, B., Li, X., Hung, T. 2014. *WPress: An Application-Driven Performance Benchmark For Cloud-Based Virtual Machines*. 2014 IEEE 18th International Enterprise Distributed Object Computing Conference. DOI 10.1109/EDOC.2014.23

Brockmeier, J. (Abril, 2012). *It's Not Highlander: There Can Be More Than One Open Source Cloud*. Consultado em Outubro, 2016, em http://readwrite.com/2012/04/06/its-not-highlander-there-can-b/

BSD License Definition, Consultado em Novembro, 2014 em http://www.linfo.org/bsdlicense.html

Buyya, R., Venugopal1, S. (2005). *A Gentle Introduction to Grid Computing and Technologies*. Computer Society of India. Disponível em http://www.cloudbus.org/papers/GridIntro-CSI2005.pdf

Buyya, R., Yeo, C., Venugopal, S. (2008). *Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities*. Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications, 5-13. DOI:10.1109/HPCC.2008.172

CA technologies. 2013. TechInsights Report: *Cloud Succeeds. Now What?* (White Paper). Consultado em Outubro, 2015, em: http://www3.ca.com/~/media/files/whitepapers/techinsights-report-cloud-succeeds.aspx

Carissimi, A. 2015. Desmistificando a Computação em Nuvem. Disponível em http://www.lbd.dcc.ufmg.br/colecoes/erad-rs/2015/002.pdf

Casos de Sucesso. Consultado em Novembro, 2014, em http://www.xpand-it.com/pt/casos-de-sucesso

Categories of free and nonfree software. Consultado em Novembro, 2014, em http://www.gnu.org/philosophy/categories.en.html

Cloud Computing – O que significam SaaS, PaaS e IaaS? Consultado em Fevereiro, 2016, em: http://www.eitisolucoes.com.br/blog/cloud-computing-o-que-significam-saas-paas-e-iaas/

Cloudbase Solutions. Consultado em Janeiro, 2014, em https://cloudbase.it/windows-cloud-images/

CloudStack. Consultado em Março, 2016, em http://docs.cloudstack.apache.org

Companies Supporting The OpenStack Foundation. Consultado em Novembro, 2016, em https://www.openstack.org/foundation/companies/

Corradi, A., Fanelli, M., Foschini, L. (2012). *VM consolidation: A real case based on OpenStack Cloud*. Future Generation Computer Systems Vol. 32, March 2014, pp. 118-127.

CSCC - Cloud Standards Customer Council. 2016. Practical Guide to Hybrid Cloud Computing. Disponível em http://www.cloud-council.org/deliverables/CSCC-Practical-Guide-to-Hybrid-Cloud-Computing.pdf

Deployment Architecture Overview. Consultado em Outubro, 2016, em http://docs.cloudstack.apache.org/en/latest/concepts.html

Debian Social Contract. Consultado em Novembro, 2014, em https://www.debian.org/social_contract

Deep Kaur, P., Chana, I. (2010). *Unfolding the Distributed Computing Paradigm*. Proceedings of the 2010 International Conference on Advances in Computer Engineering, Bangalore, India, June 2010 (Pages 339-342). IEEE Computer Society Washington

Deltacloud. Consultado em Abril, 2015, em: https://deltacloud.apache.org/about.html

Dialogic. (2010). *Introduction to Cloud Computing*. (White Paper). Consultado em Outubro, 2015, em: http://www.dialogic.com/~/media/products/docs/whitepapers/12023-cloud-computing-wp.pdf

Dongarra, J., Luszczek, P. 2001. *The LINPACK Benchmark: Past, Present, and Future*. Consultado em Abril, 2014, em http://www.netlib.org/utk/people/JackDongarra/PAPERS/hpl.pdf

Emulab. Consultado em Outubro, 2015, em: https://www.emulab.net/

Eucalyptus. Consultado em Abril, 2014, em https://www.eucalyptus.com/eucalyptus-cloud/iaas

Eucalyptus. Consultado em Agosto, 2016, em http://www8.hp.com/us/en/cloud/helion-eucalyptus.html

Fifield, T., Fleming, D., Gentle, A., Hochstein, L., Proulx, J., Toews, E., Topjian, J. (2014). *OpenStack Operations Guide: Set up and manage your openstack cloud*. O'Reilley. Disponível em: http://docs.openstack.org/ops/

Foley, J. (Setembro, 2008). *A Definition of Cloud Computing*. Consultado em Maio, 2016, em http://www.informationweek.com/cloud/a-definition-of-cloud-computing/d/d-id/1072409?

Foster, I., Zhao, Y., Raicu, I., Lu, S. (2008). *Cloud Computing and Grid Computing 360-Degree Compared*. Grid Computing Environments Workshop 2008. GCE '08, Austin, Texas. DOI: 10.1109/GCE.2008.4738445

Freeware. Consultado em Novembro, 2014, em http://www.techterms.com/definition/freeware

FSF. Consultado em Novembro, 2014, em: http://www.fsf.org/

FutureSystems. Consultado em Outubro, 2015, em https://portal.futuresystems.org/

Gartner. (Junho, 2009). *Gartner Highlights Five Attributes of Cloud Computing*. Consultado em Maio, 2016, em http://www.gartner.com/newsroom/id/1035013

Ge, X., Qi, Z., Chen, K., Duan, J., Dong, Z. 2014. *Loosely-coupled Benchmark Framework Automates Performance Modeling on IaaS Clouds*. 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. DOI 978-1-4799-7881-6/14

Gens, F. (2008). *Defining "Cloud Services" and "Cloud Computing"*. IDC eXchangeIDC. Consultado em Maio, 2016, em http://blogs.idc.com/ie/?p=190

Get started with OpenStack. Consultado em Março, 2015, em http://docs.openstack.org/

Gil, H. 2014. *A passagem da WEB 1.0 para a WEb 2.0 e... WEB 3.0: potenciais consequências para uma «humanização» em contexto educativo*. Boletim informativo. ISSN 2183-0878. Nº 5. p. 1-2. Disponível em: http://hdl.handle.net/10400.11/2404

Gillam, L., li, B., O'Loughlin, J. 2014. *Benchmarking cloud performance for service level agreement parameters*. International Journal of Cloud Computing, Vol. 3, Nº 1, 2014

GNU General Public Licence. Consultado em Novembro, 2014, em https://www.gnu.org/copyleft/gpl.html

Greenhow, C., Robelia, B., Hughes, J. (2009). *Learning, Teaching, and Scholarship in a Digital Age Web 2.0 and Classroom Research: What Path Should We Take Now?*. Educational Researcher May 2009 vol. 38 no. 4 246-259. doi: 10.3102/0013189X09336671.

Hadoop. Consultado em Outubro, 2015, em https://hadoop.apache.org/

Hewlett Packard Enterprise. (2015). *Technical White Paper: General Purpose Reference Architecture: HPE Helion Eucalyptus*. Disponível em https://www.hpe.com/h20195/V2/GetDocument.aspx?docname=4AA6-2547ENW&cc=us&lc=en

IBM Global Services. (2009). *White Paper: Cloud computing – defined and demystified*. Disponível em https://www.mercurymagazines.com/pdf/NCIBMLITSCLOUD1.pdf

Jayasinghe, D., Malkowski, S., Li, J., Wang, Q., Wang, z., Pu, C. 2013. *Variations in Performance and Scalability: An Experimental Study in IaaS Clouds Using Multi-Tier Workloads*. IEEE Transactions on Services Computing, Vol. 7, Nº 2, April-June 2014. DOI.10.1109/TSC.2013.46

KPMG India. 2011. *The Cloud: Changing the Business Ecosystem*. Disponível em https://www.kpmg.com/IN/en/IssuesAndInsights/ThoughtLeadership/The_Cloud_Changing_the_Business_Ecosystem.pdf

Kreger, H., Brunssen, V., Sawyer, R., Arsanjani, A., High, R. 2012. The IBM advantage for SOA reference architecture standards (Developer Works). Disponível em http://www.ibm.com/developerworks/library/ws-soa-ref-arch/ws-soa-ref-arch-pdf.pdf

Laszewski, G., Diaz, J., Wang, F., Fox, G. 2012. *Comparison of Multiple Cloud Frameworks*. Pervasive Technology Institute, Indiana University, Bloomington, IN 47408, U.S.A

Lista de softwares open source para Windows. Consultado em Novembro, 2014, em http://pt.wikipedia.org/wiki/Lista_de_softwares_open_source_para_Windows

Llorente, I. M. (Fevereiro, 2013). *Eucalyptus, CloudStack, OpenStack and OpenNebula: A Tale of Two Cloud Models*. Consultado em Outubro, 2016, em http://opennebula.org/eucalyptus-cloudstack-openstack-and-opennebula-a-tale-of-two-cloud-models/

Marston, S., Li, Z., Bandyopadhyay, S., Ghalsasi, A. (2011). *Cloud Computing - The Business Perspective*. 44th Hawaii International Conference on System Sciences (HICSS). DOI 10.1109/HICSS.2011.102

Maverick, J. 2015. *What is the difference between CAPEX and OPEX?* Disponível em http://www.investopedia.com/ask/answers/020915/what-difference-between-capex-and-opex.asp

Mell, P. Grance, T. (2011). *The NIST definition of cloud computing, Recommendation of the national institute of standards and technology*. NIST special publication 800-145, National Institute of Standards and Technology, U.S. Department of Commerce.

Montero, R., Moreno-Vozmediano, R., Llorente, I. (2012). *IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures*. Computer, vol. 45, no. , pp. 65-72, Dec. 2012, doi:10.1109/MC.2012.76

Nimbus. Consultado em Abril, 2015, em: http://www.nimbusproject.org/about/

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Youseff, L., Zagorodnov, D. 2009. *The Eucalyptus Open-source Cloud-computing System*. Disponível em http://www.cs.ucsb.edu/~rich/publications/ccgrid09.pdf

Official Documentation for Eucalyptus Cloud. Consultado em Abril, 2016, em http://docs.hpcloud.com/eucalyptus/4.2.2/#shared/index.html

Open Cloud Architecture. Consultado em Outubro, 2016, em http://docs.opennebula.org/5.0/deployment/cloud_design/open_cloud_architecture.html

Open Source. Consultado em Outubro, 2014, em http://opensource.org/about

OpenNebula. Consultado em Abril, 2015, em: http://opennebula.org/about/

OpenNebula Systems. 2016. *OpenNebula 5.0 Introduction and Release Notes Release 5.0.2*. Disponível em http://docs.opennebula.org/pdf/5.0/opennebula_5.0_intro_release_notes.pdf

OpenStack. Consultado em Setembro, 2014, em https://www.openstack.org/

OpenStack Havana Architecture Overview. Consultado em Março, 2014, em https://wiki.openstack.org/wiki/ArchitecturalOverview

OpenStack Releases. Consultado em Setembro, 2016, em https://releases.openstack.org/

Paradowski, A., Liu, L., Yuan, B. 2014. *Benchmarking the Performance of OpenStack and CloudStack*. 2014 IEEE 17th International Symposium on Object/Component-Oriented Real-Time Distributed Computing. DOI 10.1109/ISORC.2014.12

Plugaru, V., Varrette, S., Bouvry, P. 2014. *Performance Analysis of Cloud Environments on Top of Energy-Efficient Platforms Featuring Low Power Processors*. 2014 IEEE 6th

International Conference on Cloud Computing Technology and Science. DOI 10.1109/CloudCom.2014.94

Plummer, D., Bittman, T., Austin, T., Cearley, D., Smith, D. (2008). *Cloud computing: Defining and describing an emerging phenomenon*. Gartner Research ID Number: G00156220. Disponível via ResearchGate em: https://www.researchgate.net/publication/265182636_Cloud_Computing_Defining_and_Describing_an_Emerging_Phenomenon

Porting Windows to OpenStack. Consultado em Julho, 2015, em https://poolsidemenace.wordpress.com/2011/06/16/porting-windows-to-openstack/

Qevani, E., Panagopoulou, M., Stampoltas, C., Tsitsipas, A., Kyriazis, D., Themistocleous1, M. 2014. *What can OpenStack adopt from a Ganeti-based open-source IaaS?* 2014 IEEE International Conference on Cloud Computing. DOI 10.1109/CLOUD.2014.115

Raymond, E. (1997). The Cthedral and the Bazaar, Linux Kongress, Wurzburg, Germany.

Rally. Consultado em Setembro, 2015, em http://docs.openstack.org/developer/rally/

Sandoval, K. (2015, Julho 8) *Living in the Cloud Stack – Understanding SaaS, PaaS, and IaaS APIs*. [Mensagem de blog] Disponível em http://nordicapis.com/living-in-the-cloud-stack-understanding-saas-paas-and-iaas-apis/

RUBBoS. Consultado em Abril, 2014, em http://jmob.ow2.org/rubbos.html

Shapeblue. Apache CloudStack. Consultado em Novembro, 2016, em http://www.shapeblue.com/pt-br/apache-cloudstack/

Shareware. Consultado em Novembro, 2014, em http://www.techterms.com/definition/shareware

Sinfic, *O que é uma solução open source?*, Sinfic SA, 2008, Consultado em Novembro, 2014, em http://www.sinfic.pt/SinficWeb/displayconteudo.do2?numero=24957

Sobel, W., Subramanyam, S., Sucharitakul, A., Nguyen, J., Wong, H., Klepchukov, A., Patil, S., Fox, A., Patterson, D. (2008). *Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0*. In Cloud Computing and Its Applications Proceedings of the 1st Workshop on Cloud Computing

Steinmetz, D., Perrault, B., Nordeen, R., Wilson, J., Wang, Xinli. 2012. *Cloud Computing Performance Benchmarking and Virtual Machine Launch Time*. SIGITE'12, October 11–13, 2012, Calgary, Alberta, Canada. ACM 978-1-4503-1464-0/12/10

Steyn, W. (2012). *Open Source Software – What are the Risks?*, Al Tamimi & Company.

STREAM – Open Multi-Processing Benchmark. Consultado em Abril, 2014, em: https://www.cs.virginia.edu/stream/ref.html

Sultan, N. (2011). *Reaching for the "cloud": How SMEs can manage*. International Journal of Information Management, Volume 31, Issue 3, 272–278

The Apache Software Foundation. Consultado em Novembro, 2014, em http://www.apache.org/licenses/LICENSE-2.0.html

The VGrADS Project. Consultado em Abril, 2016, em http://vgrads.rice.edu/

Using Bonnie++ for filesystem performance benchmarking. Consultado em Abril, 2014, http://archive09.linux.com/feature/139742

Varghese, B., Akgun, O., Miguel, I., Thai, L., Barker, A. 2014. *Cloud Benchmarking for Performance*. 2014 IEEE 6th International Conference on Cloud Computing Technology and Science. DOI 10.1109/CloudCom.2014.28

What is a Public Cloud? Discover Top Rated Public Cloud Computing Providers, Services, Security & Technologies. Consultado em Dezembro, 2015, em: http://cloudnewsdaily.com/public-cloud/

What is Copyleft. Consultado em Novembro, 2014, em http://www.gnu.org/copyleft/

Windows guests - build ISOs including VirtIO drivers. Consultado em Março, 2014, em https://pve.proxmox.com/wiki/Windows_guests_-_build_ISOs_including_VirtIO_drivers

Windows VirtIO Drivers. Consultado em Março, 2014, em http://www.linux-kvm.org/page/WindowsGuestDrivers/Download_Drivers

Xen Cloud Platform. Consultado em Abril, 2015, em: http://www-archive.xenproject.org/products/cloudxen.html

Xu, Q., Yuan, J. 2014. *A Study on Service Performance Evaluation of Openstack*. 2014 Ninth International Conference on Broadband and Wireless Computing, Communication and Applications. DOI 10.1109/BWCCA.2014.120

Zero to Cloud Guide. Consultado em Novembro, 2016, em http://www.nimbusproject.org/docs/2.10.1/admin/z2c/index.html

Zhang, S., Zhang, S., Chen, X., Huo, X. (2010). *Cloud Computing Research and Development Trend*. 2010 Second International Conference on Future Networks. DOI 10.1109/ICFN.2010.58

ANEXO A

"A low cost private cloud infrastructure using OpenStack" publicado na conferência IEEE BigData 2014 Coimbra Satellite Session

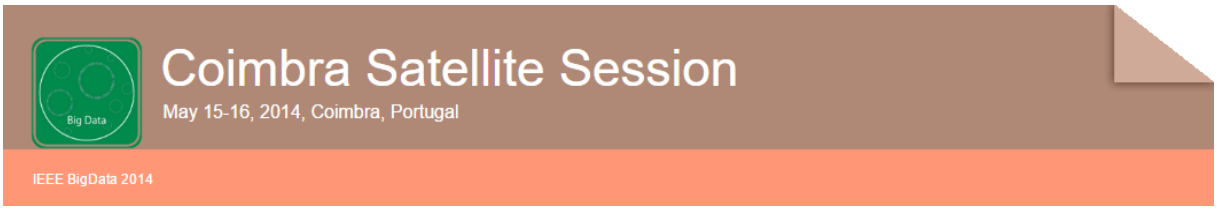

Figura A.1 - Logotipo da conferência *IEEE BigData 2014 Coimbra Satellite Session*

A low cost private cloud infrastructure using Openstack

Tiago Rosado
Polytechnic Institute of Coimbra
ISEC
Coimbra, Portugal
a20105059@alunos.isec.pt

Jorge Bernardino
Polytechnic Institute of Coimbra
ISEC
Coimbra, Portugal
jorge@isec.pt

***Abstract*—Cloud computing solutions evolved and promise to deliver to the end user resources like hardware, network or storage in form of services at a low cost with an easy to use interface over the Internet. This paper gives an overview of cloud computing and outlines an effort for providing guidelines for enterprises of how to deliver a feasible and reliable low cost cloud infrastructure as a service for technological environment using Openstack.**

Keywords–openstack; cloud computing; private cloud computing; open source; low cost; infrastructure as a service.

I. INTRODUCTION

Cloud computing architectures have gained more and more attention in recent years and the term is often applied to an elastic computational model where a diverse array of IT resources and services are delivered to users through a broad network connection as Internet.

As a result and a consequence of the ease of access to the Internet, this new concept satisfies an increasing need of IT services on demand without a direct investment in new infrastructure, training new personnel or software licensing, thus reducing the time to market and facilitating a higher pace of innovation.

Enterprise IT Organizations focused on the cutting edge of technology, developing new business solutions and platforms, have a constant need of resources for supporting their activity. Those resources could be storage, quality assurance servers or test beds and will imply different and specific configurations according to each project, development team or department. Rely on the IT department for bare metal delivery and the provisioning of such variety, specific and constant changes in configurations for each server or infrastructure, consumes time the teams do not have. While IT struggles to provide the resources necessary to make services available and meet today's demands, management board are focused on controlling costs and investments.

On the premise that improvement comes from inside, this work aims to deliver an open source private cloud computing implementation in order to respond IT Organizations demands regarding cost control issues, rapid and highly customizable infrastructure, maintaining high standards of quality, security and robustness. Implementing an open source private cloud computing solution avoids vendor lock-in and will provide resources on-demand to research and development departments well as independence to manage them without jeopardizes the enterprises critical IT infrastructure.

There are different offerings of cloud solutions like infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS). There are many studies about infrastructure as a service (IaaS) solutions focused on the study of middleware platforms and on comparative studies [9][10][11] of the different open source solutions like Openstack [12][13], Eucalyptus [14] or OpenNebula [15], providing users a good basis in order to achieve freedom of choice. Studies on Openstack have shown major qualities focusing scalability and modularity [17], security [6] [7] and good performance in high demands [16]. However Openstack evolved since then having grown the partnership of big enterprises on the support and development, requiring an updated study.

In this paper our focus is in the infrastructures, addressing a suitability of such implementation using the open source cloud provider solution from Openstack for maximizing existing in-house resources.

The remainder of this paper is organized as follows. Section 2 gives an overview of what cloud computing is, followed by information of cloud service and deployment models. Section 3 gives detailed information of the design and implementation of an infrastructure as a service in Openstack. Lastly, section 4 presents our conclusions and future work.

II. CLOUD COMPUTING OVERVIEW

In this section we give an overview of cloud computing area beginning with its definition and main characteristics. Following, we describe the three service models: IaaS, PaaS, and SaaS. Finally the three major delivery models offered by cloud computing are explained: private cloud, public cloud and hybrid cloud.

A. Definition

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction, as stated in [1].

The focus of cloud computing is on architectures, security and deployment strategies, basing their strengths on five key aspects [1] [8]:

- *On-demand self-service*: Giving the consumer the possibility to quickly develop and deploy applications or manage computer resources without prohibitive start-ups costs or relying on third-party.
- *Ubiquitous network access*: Resources are available through internet-enabled platforms like mobile phones, tablets, laptops or thin clients.
- *Resource pooling*: The multiple physical and virtual resources are pooled to serve multiple user demands, abstracting them from the physical locations of those resources.
- *Elasticity*: Gives the appearance of unlimited resources due to their provisioning, in some cases automatically, to face up-scaling or down-scaling system demands for rapidly respond to new business and operational requisites.
- *Measuring and Monitoring*: Leads the infrastructure to a near constant uptime due to constant maintenance and proactive response to system alerts.

B. *Service Models*

The leverage on cloud computing for end users implies they do not need to manage the underlying fixed infrastructure used to provide cloud services, thereby; those resources are delivered to users through three major service models.

1) Infrastructure as a Service (IaaS)

Is a standardized service model that delivers computer infrastructure (storage, data center space and networking capabilities) owned and supported by a service provider and delivered to customers on-demand to support enterprise operations. Resources can be accessed for management using a web-based graphical user interface [2] [3].

2) Platform as a Service (PaaS)

Is a service model that delivers a broad collection of application infrastructure services allowing users deploying personalized applications. In this model, unlike IaaS, user does not have a direct access to the systems resources. Examples of this would include Microsoft's Azure or Google's App Engine [4].

3) Software as a Service (SaaS)

Is a software delivery model where an application is hosted on a remote data center and provided as a service to the final consumer across the Internet. It's software that is rented rather than purchased. Examples of this would include Microsoft's Office 365 or Google App's.

C. *Delivery Models*

According to the deployment style and service object there are three major delivery models offered by cloud computing: private cloud, public cloud and hybrid cloud.

1) Private cloud

A private or internal cloud applies the concepts of a cloud based environment where all resources (physical and virtualized infrastructure) are wholly owned and hosted within the enterprise's IT environment who is consuming the service. The pool of resources is only accessible and managed by a single organization. The aim of a private cloud implementation can be for maximizing existing in-house resources, security and privacy concerns regarding critical core business or data transfer ratios.

2) Public cloud

In spite of concept being the same, in a public cloud environment, final user only takes advantage from the service provided whether it is infrastructure, platform or software. The cloud provider owns pools of shared physical resources providing services to multiple clients where is applied its own policy, profit and billing model.

3) Hybrid cloud

Hybrid is a cloud environment whose concept combines both private and public clouds, bound together by standardized or proprietary technology, providing distinct functions within the same organization. Organizations apply this model in order to reach optimization by sending peripheral business functions to the public cloud and maintain core activities in-house.

III. Design and Implementation of an Infrastructure as a Service in Openstack

Openstack is a fully open sourced cloud solution, released under the terms of the Apache license, for delivering and managing Cloud IaaS originally developed by NASA and Rackspace. At this time is governed by The Openstack Foundation having as supporter's companies like Canonical, Suse, Red Hat, Nebula, HP, IBM, Rackspace, Intel, AMD, Cisco, among others. By being open source and free to download gives to small players the possibility to deploying small cloud infrastructures. Openstack is a combination of software projects considered one of the most complete solutions with great potential due to its architecture, community and partner's support.

A. *Openstack architecture*

Openstack solution can be divided into three major groups of services namely Computing, Networking and Storing, each of them implements services with different functionalities [12] [18]. By being a series of interrelated projects provides users freedom to better plan an architecture to fit their purpose, for instance, having one dedicated server for a specific function in the infrastructure in other words, each server only has installed the services needed for the intended design.

Computing

- Openstack Compute, also known as Nova or Nova Compute provides virtual servers upon demand. This is a management platform that manages the cloud resources through application programming interfaces (API's) who interact with the hypervisors. Nova Compute supports several kinds of hypervisors like KVM, Xen, VMware, or Hyper-V.

- Openstack Image service also known as Glance is a lookup and retrieval system for virtual machine (VM) images. It provides services for discovering, registering and retrieving virtual images through an API that allows querying of VM image metadata.

Networking

- Openstack Networking also known as Neutron is a project to provide network connectivity as a service between interface devices managed by other Openstack services (e.g. Nova). The service works by allowing users to create their own networks and then attach interfaces to them. It provides capabilities for managing static internet protocols (IP's), dynamic host configuration protocol (DHCP) or virtual local area networks (VLAN's). Its pluggable architecture let's user's take leverage on frameworks (e.g. intrusion detection systems, load balancing, virtual private networks) from supported vendors.

Storing

- Openstack Object Storage also known as Swift is a highly available, distributed object / blob store. Allows users to store or retrieve files.
- Openstack Block Storage also known as Cinder or Cinder Volumes. Provides persistent block storage (or volumes) to guest VMs. In earlier releases of Openstack this service was an integral part of Nova in the form of nova-volume. Cinder will mainly interact with Nova, providing volumes for its instances, allowing through its API the manipulation of volumes, volume types and volume snapshots.

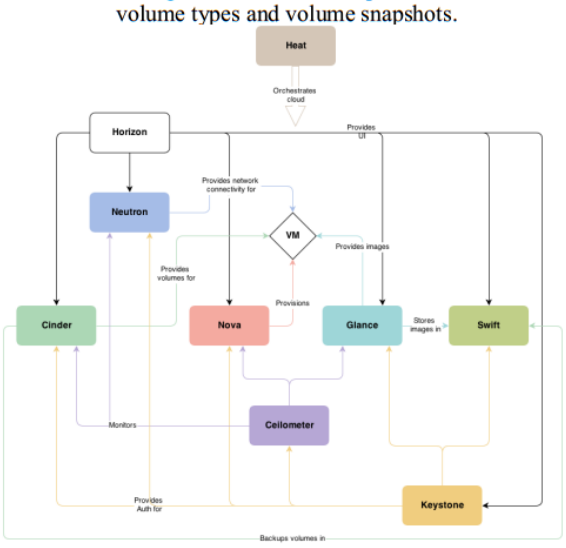

Figure 1 - Opensatck conceptual architecture

Figure 1 represents the Openstack conceptual architecture with all the existing services for this solution and the interactions between them, highlighting two central services. Those two main services interacts with all the others in the architecture whatever it may be, are they, Horizon and Keystone, the Openstack dashboard and the Openstack identity respectively.

Openstack dashboard: Also known as Horizon is a modular web application that provides an interface for cloud infrastructure management by interacting with all other services public APIs.

Openstack identity: Also known as Keystone is a single point of integration for Openstack policy, catalog, token and authentication applying them to users and services interactions. Without keystone there's no compliance between services consequently the cloud solutions won't be available for end users.

B. *Use Case*

In this implementation we used Ubuntu Server 12.04 Long Term Support (LTS) for its reliability regarding longer support and maintenance and Havana has the last stable release of Openstack. Aiming to provide a clear, implementable, modular and yet still performant solution was used commercial range hardware.

For easier understanding, Openstack uses the term Cloud Controller to identify the main node in the infrastructure and Compute Nodes for all the others. The technical main characteristics are the following:

- Cloud controller – Intel® Core™ i3-3220 CPU @ 3.30GHz, 4 GB DDR3 / 1067 MHz, 250Gb Hard Disc Drive. Added one extra Gigabit Ethernet controller.
- Compute Node – Intel® Core™ i3-3220 CPU @ 3.30GHz, 16 GB DDR3 / 1067 MHz, 250Gb Hard Disc Drive. Added one extra Gigabit Ethernet controller
- Networking – Two eight port Gigabit Ethernet switches

All the nodes in the infrastructure communicate via two Gigabit private switches, one for VM's internal communications and another to ensure a connection to the enterprise internal network. The controller node is configured to do all the backend processes via Nova, image management via Glance, hosting the website via Horizon and dealing with user identification via Keystone. All the compute nodes are configured with the Nova compute services and kernel-based virtual machine infrastructure (KVM). The network layout and all the physical connections for the cloud implementation as described can be seen in Figure 2.

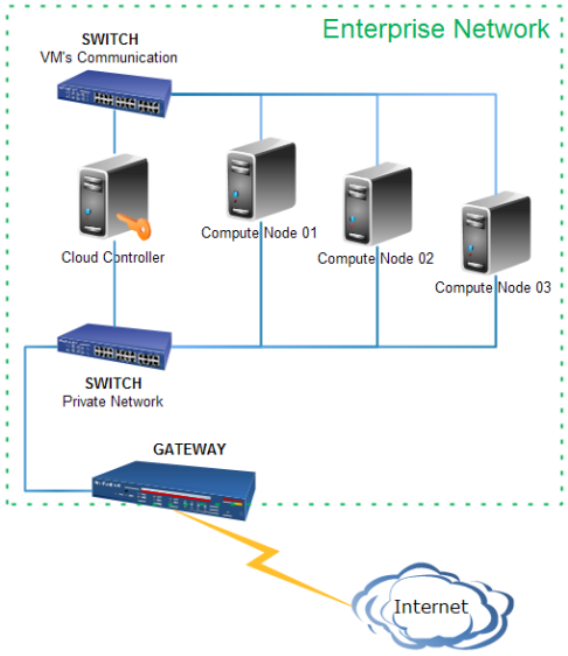

Figure 2 - Cloud Network Topology

IV. CONCLUSIONS AND FUTURE WORK

The cloud computing paradigm, such as Openstack with a big user base in academy and industry has a great potential to provide flexible, reliable and rapid resources on demand, leveraging from today's easy access to a broad network connection.

This work has presented an overview of cloud computing as well as a specific insight over Openstack architecture and how it adapts not only to major data centers but also into commercial hardware, providing a sustainable and robust private cloud solution.

Using open source software such as Openstack, with major tests given, on commercial range hardware enables enterprises to deliver an Infrastructure as a Service to research and development departments well as reduction in costs with software licensing and time spent on server's creation, configuration and maintenance.

Based on this study, future work will involve an investigation in methodologies to measure and monitor activities in order to evolve configuration towards a more efficient and effective solution.

ACKNOWLEDGMENTS

The authors would like to thank Paulo Rosado from the Mobile Device Management Department at the PT Cloud e Data Centers SA and Délio Lourenço from the IS/IT Department at PT Comunicações for all the support. The authors would like to acknowledge the company Portugal Telecom the availability to practical application the design of experiment. We also would like to thank Instituto Superior de Engenharia de Coimbra (ISEC) for its support in this work.

REFERENCES

[1] P. Mell, T. Grance, "The NIST definition of cloud computing, recommendation of the national institute of standards and technology", NIST special publication 800-145, National Institute of Standards and Technology, U.S. Department of Commerce, Sept 2011.

[2] Infrastructure as a Service – [online] http://www.gartner.com/it-glossary/infrastructure-as-a-service-iaas (Accessed February 2014).

[3] Konstantinos Kostantos, Andrew Kapsalis, Dimosthenis Kyriazis, Marinos Themistocleous, Paulo Rupino da Cunha. (2013) *Open-source IaaS Fit for Purpose: A Comparison between Opennebula and Openstack*. International Journal of Electronic Business Management, Vol. 11, No. 3, pp. 191-201, 2013.

[4] Platform as a Service – [online] http://www.gartner.com/it-glossary/platform-as-a-service-paas/ (Accessed February 2014).

[5] Software as a Service – [online] http://www.techterms.com/definition/saas (Accessed February 2014)

[6] Y. Chen, V. Paxson, R. H. Katz, "What's New About Cloud Computing Security?", Electrical Engineering Computer Sciences, University of California at Berkeley, Technical Report No. UCB/EECS-2010-5, Jan. 2010.

[7] A. TaheriMonfared, M. G. Jaatun, "As strong as the weakest link: Handling compromised components in OpenStack", 2011 Third IEEE International Conference on Cloud Computing Technology and Science, 2011.

[8] R. Dargha, "Cloud computing: from hype to reality. Fast tracking cloud adoption", International Conference on Advances in computing, Comunications and Informatics 2012, Aug 2012.

[9] G.Laszewski, J. Diaz, F.Wang, G. C. Fox, "Comparison of multiple cloud frameworks" Pervasive Technology Institute, Indiana University, 2012.

[10] M. Bist, M. Wariya, A. Agarwal, "Comparing Delta, Open Stack and Xen Cloud Platforms: A Survey on Open Source IaaS", 2013 3rd IEEE International Advance Computing Conference, 2013.

[11] X. Wen, G. Gu, Q. Li, Y. Gao, X. Zhang, "Comparison of open-source cloud management platforms: OpenStack and OpenNebula", 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, 2012

[12] Openstack – [online] http://www.openstack.org/ (Accessed March 2014)

[13] O. Sefraoui, M. Aissaoui, M. Eleuldj, "OpenStack: Toward an Open-Source Solution for Cloud Computing", International Journal of Computer Applications (0975-8887) Volume 55 Nº 03, October 2012.

[14] Eucalyptus – [online] https://www.eucalyptus.com (Accessed March 2014)

[15] OpenNebula – [online] http://opennebula.org/ (Accessed March 2014)

[16] D. Steinmetz, B. Perrault, R. Nordeen, J. Wilson, X. Wang, "Cloud Computing Performance Benchmarking and Virtual Machine Launch Time", SIGITE'12 Calgary, Alberta, Canada.

[17] J. A. L. Castillo, K. Mallichan, Y. Al-Hazmi, "Openstack federation in experimentation multi-cloud testbeds", 2013 IEEE International Conference on Cloud Computing Technology and Science, 2013.

[18] Deploying Openstack – [online] http://ken.pepple.info/ (accessed April 2014)

ANEXO B

"An overview of OpenStack architecture" publicado na conferência 18th International Database Engineering & Applications Symposium, IDEAS '14P

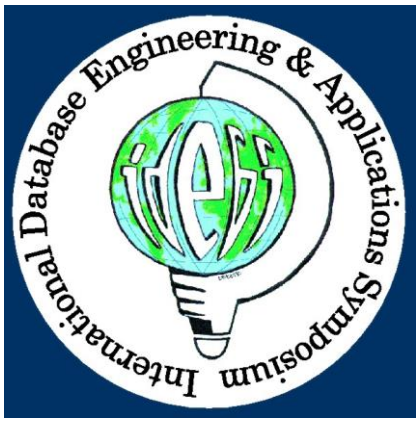

Figura B.1 - Logotipo da conferência *18th International Database Engineering & Applications Symposium, IDEAS '14P*

An Overview of Openstack Architecture

Tiago Rosado
Polytechnic Institute of Coimbra
ISEC – Coimbra Institute of Engineering
Rua Pedro Nunes, 3030-199 Coimbra, Portugal
Tel. ++351 239 790 200
a20105059@alunos.isec.pt

Jorge Bernardino
Polytechnic Institute of Coimbra
ISEC – Coimbra Institute of Engineering
Rua Pedro Nunes, 3030-199 Coimbra, Portugal
Tel. ++351 239 790 200
jorge@isec.pt

ABSTRACT
Cloud Computing concept refers to both the applications delivered as services over the Internet and the servers and system software in the datacenters that provide those services. These solutions offer pools of virtualized computing resources, paid on a pay-per-use basis, and drastically reduce the initial investment and maintenance costs. Efficient and flexible resource management is the main focus for the cloud solutions on the market as well as scalability and adaptability to new environments. Openstack exceeded the market as a scalable, performant and highly adaptive open source architecture for both public and private cloud solutions as well as leveraging from hardware resources either they be professional or entry level. This paper gives an overview of Openstack software components functionalities in order to design and implement unique cloud computing solutions to fit enterprises purposes.

Categories and Subject Descriptors
C.5 [**Computer System Implementation**]: Miscellaneous
D.2.11 [**Software Engineering**]: Software Architectures – Domain-specific architectures

General Terms
Design, Experimentation, Performance.

Keywords
Openstack, IaaS, Cloud Computing, Open Source.

1. INTRODUCTION
Cloud computing have gained more and more attention in recent years, and open source solutions in general have been growing in terms of quality, security and internal requirements. Leading the trend on open source in the IT scenarios are the cloud computing platforms. Cloud computing is a model for delivering ubiquitous, on-demand computer resources over the Internet, relying on essential characteristics as on-demand self-service, broad network access, resource pooling, elasticity, measuring and monitoring. All resources like storage, computing or network are presented to final users as services in the form of infrastructure, platform or software. This emerging model can be applied inside enterprises, known as private clouds, or acquired as services form external vendors, called public clouds.

IDEAS '14, July 07 - 09 2014, Porto, Portugal

http://dx.doi.org/10.1145/2628194.2628195

OpenStack is the open source cloud computing platform most widely adopted in industry. Meant to be simple to implement, this Infrastructure as a Service (IaaS) is open and massively scalable. There are many studies focusing on open source solutions for cloud computing, presenting detailed comparisons and overviews between them [5] [1]. Others exist pointing studies on modules in particular [6] or detailed success implementations of Openstack on education and industry [2][3]. This paper gives an update, highlight and detailed overview of Openstack architecture, showing the essential services that are necessary to install.

2. OPENSTACK ARCHITECTURE
Openstack is a fully open sourced combination of software projects developed originally by NASA and Rackspace, having gained so far such reliability robustness and availability as an IaaS solution for both public and private cloud infrastructures [4]. Its modular and highly configurable architecture enables this solution to be conceived and tailored to fit available hardware resources either they be professional or entry level, with few or more computer nodes, in response to each case in particular. This allows enterprises to choose from a variety of complementary services in order to meet different needs regarding to computing, networking and storage [7]. Figure 1 represents the Openstack conceptual architecture with all native software components, developed by companies and individual supporters, depicting how they interact with each other.

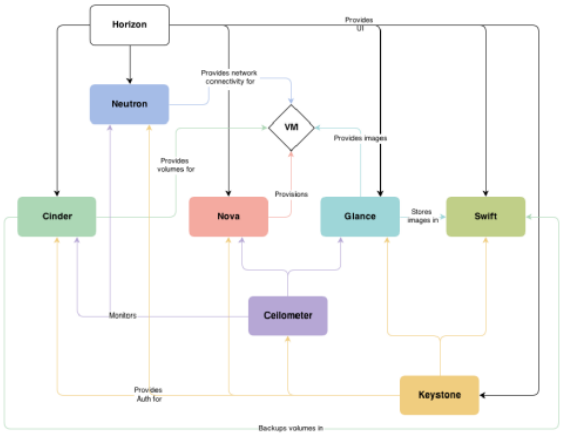

Figure 1 - Openstack conceptual architecture

Following we describe those components dividing them into five groups due to their characteristics such as computing, networking, storing, shared and supporting services. Although the supporting services are not native from Openstack's core, are essential for its

operation [8]. Services can be installed in accordance with requirements, which mean that we can install all or only a few.

2.1 Computing

Nova: Also known as Openstack Compute or Nova Compute, it provides virtual servers upon demand interacting with the hypervisors, supporting several kinds such as KVM, Xen, VMware or Hyper-V. It's an array of software that provides services for cloud resource management through its APIs, capable of orchestrating running instances, networks and access control. It's an essential service for a basic cloud architecture implementation.

Glance: It's the Openstack Image service, a lookup and retrieval system for virtual machine (VM) images. It provides services for discovering, registering and retrieving virtual images through an API that allows querying of VM image metadata, catalog and manage large libraries of server images. It's an essential service for a basic cloud architecture implementation.

2.2 Networking

Neutron: The Openstack networking services is a project to provide network connectivity as a service between interface devices managed by other Openstack services (e.g. Nova). It provides capabilities for managing dynamic host configuration protocol (DHCP), static internet protocols (IP's), or virtual area networks (VLAN's) along with other advanced policies and topologies. Due to its architecture it allows users to take leverage of frameworks (e.g. intrusion detection systems, load balancing, virtual private networks) from supported vendors.

2.3 Storing

Swift: Also known as Openstack Object Storage, Swift is a highly available, distributed object/blob store. Allows users to store or retrieve files. It can be used by Cinder component to back up the VMs volumes.

Cinder: Also known as Openstack Block Storage or Cinder Volumes it provides persistent block storage (or volumes) to guest virtual machines. By working with Swift, Cinder can use it to back up the VMs volumes. In earlier releases of Openstack this service was an integral part of Nova in the form of nova-volume. Cinder will mainly interact with Nova, providing volumes for its instances, allowing through its API the manipulation of volumes, volume types and volume snapshots.

2.4 Shared Services

Keystone: The Openstack identity is a single point of integration for Openstack policy, catalog, token and authentication, applying them to users and services interactions. Without Keystone there's no compliance between services consequently the cloud solutions won't be available for end users. It's an essential service for a basic cloud architecture implementation.

Horizon: The Openstack dashboard is a modular web application that provides a user interface for cloud infrastructure management by interacting with all other services public APIs. It's an essential service for a basic cloud architecture implementation.

Ceilometer: Is a new component of the Openstack services that provides a configurable collection of metering data in terms of CPU and network costs available from all other services in the platform, delivering a unique point of contact for billing systems.

2.5 Supporting Services

Database: By default Openstack uses MySQL as its relational database management system for core services to store configurations and management information. Typically MySQL database is installed on the main node of the infrastructure, the controller node. It's an essential service for a basic cloud architecture implementation.

Advanced Message Queue Protocol: AMQP is the messaging technology used in Openstack for inter-process communication. It uses by default RabbitMQ and supports others such as Qpid or ZeroMQ. The broker used gives a common platform for processes to send and receive messages between them making use of little or no knowledge of each other's component definitions. It's an essential service for a basic cloud architecture implementation.

3. CONCLUSIONS AND FUTURE WORK

Cloud architectures exploit virtualization techniques to provision multiple Virtual Machines (VMs) on the same physical host, so as to efficiently use available resources. The cloud computing paradigm and available solutions in the market, regarding to Openstack, have been implemented and explored in various case scenarios, with major tests given in academy and industry, presenting good results for performance, reliability and final user acceptance. Openstack leverages due to its modular architecture, simplicity to implement and adaptation not only to major data centers but also to entry level hardware providing enterprises a massively scalable cloud infrastructure. Allied to the fact of being open source, which helps on cost control, enables to deliver rapidly on-demand resources to final users. This work has presented an overview of Openstack architecture, services available and how they work and adapt into any hardware infrastructure providing a sustainable and robust private cloud solution. Based on this study, future work will involve an investigation on how to improve final user experience with the platform and administration activities regarding to measures and monitoring the cloud environment.

4. REFERENCES

[1] Bist, M. Wariya, M. Agarwal, A. 2012. Comparing Delta, Open Stack and Xen Cloud Platforms: A Survey on Open Source IaaS. In 2013 3rd IEEE International Advance Computing Conference (IACC).

[2] Bonner, S. Pulley, C. Kureshy, I. Holmes, V. Brennan, J. James, Y. 2013. Using OpenStack to Improve Student Experience in an H.E. Environment. In Science and Information Conference 2013, London, UK, October 2013.

[3] Campos, I. Fernández-del-Castillo, E. Heinemeyer, S. Lopez-Garcia, A. Pahlen, F.v.d. 2013. Borges, G. Phenomenology tools on cloud infrastructures using OpenStack. The European Physical Journal C.

[4] Castillo, J.Á.L.d. Mallichan, K. Al-Hazmi, Y. OpenStack Federation in Experimentation Multi-cloud Testbeds. 2013. In 2013 IEEE ICCCTS.

[5] Laszewski, G.v. Diaz, J. Wang, F. Fox, G. C. 2012. Comparison of Multiple Cloud Frameworks. Pervasive Technology Institute, Indiana University. Bloomington, IN 47408, U.S.A.

[6] Litvinski, O. Gherbi, A. Experimental Evaluation of Openstack Compute Scheduler. 2013. In The 4th International Conference on Ambient Systems, Networks and Technologies.

[7] Openstack – [online] http://www.openstack.org/ (Accessed April 2014).

[8] Pepple, K. 2013. *Deploying Openstack, Creating open source clouds, 2nd Edition*, O'Reilly Media.

ANEXO C

“Implementation of a Low Cost IaaS using OpenStack” publicado na conferência 11th International Joint Conference on Software Technologies (ICSOFT 2016)

Figura C.1 - Logotipo da conferência *11th International Joint Conference on Software Technologies (ICSOFT 2016)*

Implementation of a low cost IaaS using Openstack

Tiago Rosado[1], Jorge Bernardino[1,2]
[1] *ISEC – Superior Institute of Engineering of Coimbra*
Polytechnic of Coimbra, 3030-190 Coimbra, Portugal
[2] *CISUC – Centre of Informatics and Systems of the University of Coimbra*
University of Coimbra, 3030-290 Coimbra, Portugal
a20105059@alunos.isec.pt, jorge@isec.pt

Keywords: Openstack, IaaS, Cloud Computing, Open Source, Low Cost.

Abstract: Cloud computing has emerged as an important paradigm enabling software, infrastructure, and information to be used as services over the network in an on-demand manner. Cloud computing infrastructures can provide adaptive resource provisioning with very little initial investment while scaling to a large number of commodity computing nodes. In this paper, we present Openstack, a modular and highly adaptive open source architecture for both public and private cloud solutions. It is shown an experimental implementation of an IaaS, built on entry-level hardware, demonstrating the hardware, and most important, the basic software components needed to set the foundation for a solid entry-point to setup a low-cost Openstack cloud infrastructure.

1 INTRODUCTION

Cloud computing has emerged as a cost-effective and elastic computing paradigm for massively scalable, fault-tolerant, and adaptive computation. Cloud computing architectures scale to large numbers of commodity computers and adapt to changing hardware availability and requirements by dynamically allocating virtualized computing nodes. One of the main advantages of the cloud computing paradigm is that it simplifies the time-consuming processes of hardware provisioning, hardware purchasing and software deployment. For example, by decoupling the management of the infrastructure (cloud providers) from its use (cloud tenants), and by allowing the sharing of massive infrastructures, cloud computing delivers unprecedented economical and scalability benefits for existing applications and enables new scenarios. This comes at the cost of increased complexity in managing a highly multi-tenant infrastructure and posing new questions on attribution, pricing, isolation, scalability, fault-tolerance, load balancing, etc.

Cloud infrastructures can provide adaptive resource provisioning with very little initial investment while scaling to massive amounts of commodity computing nodes. Enterprise IT Organizations focused on the cutting edge of technology, developing new business solutions and platforms, have a constant need of resources for supporting their activity. Those resources could be storage, quality assurance servers or test beds and will imply different and specific configurations according to each project, development team or department. Rely on the IT department for delivery and the provisioning of such variety, changes in configurations for each server or infrastructure, consumes much time and human resources.

On the premise that improvement comes from inside, this work aims to deliver an open source private cloud computing implementation in order to respond IT Organizations demands regarding cost control issues, rapid and highly customizable infrastructure, maintaining high standards of quality, security and robustness. Implementing an open source private cloud computing solution avoids vendor lock-in and will provide resources on-demand to research and development departments well as independence to manage them without jeopardizes the enterprises critical IT infrastructure.

There are different offerings of cloud solutions like Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). There are many studies about IaaS solutions focused on the study of middleware platforms and on comparative studies (Laszewski, 2012) (Bist, 2013) of the different open source solutions like Openstack (Openstack, 2015) (Sefraoui, 2012), Eucalyptus (Eucalyptus, 2015) or OpenNebula (OpenNebula, 2015), providing users a good basis in order to achieve freedom of choice. Studies on Openstack have shown major qualities focusing scalability and modularity (Castillo, 2013), security (Chen, 2010) (TaheriMonared, 2011) and good performance in high demands (Steinmetz, 2012). However Openstack evolved since then having grown the partnership of big enterprises on the support and development, requiring an updated study. In this paper our focus is in the infrastructures, addressing a suitability of such implementation using the open source cloud provider solution from Openstack for maximizing existing in-house resources.

The remainder of this paper is organized as follows. Section 2 presents Openstack architecture and service followed by the object model. Section 3 presents a use case of how to design and implement an IaaS using Openstack. Finally, conclusions and future work are given in Section 4.

2 OPENSTACK ARCHITECTURE

Openstack is a fully open sourced cloud solution, released under the terms of the Apache license, for delivering and managing Cloud IaaS originally developed by NASA and Rackspace(Castillo, 2013). At this time is governed by The Openstack Foundation having as supporter's companies like Canonical, Suse, Red Hat, Nebula, HP, IBM, Rackspace, Intel, AMD, Cisco, among others. By being open source and free to download gives to small players the possibility to deploying small cloud infrastructures. Openstack is a combination of software projects considered one of the most complete solutions with great potential due to its architecture, community and partner's support having gained so far such reliability robustness and availability as an IaaS solution for both public and private cloud infrastructures (Castillo, 2013).

2.1 Conceptual Architecture

The modular and highly configurable architecture of Openstack enables this solution to be conceived and tailored to fit available hardware resources. It can be used by professional or entry level users, with small or large number of computer nodes. This allows enterprises to choose from a variety of complementary services in order to meet different needs, regarding computing, networking and storage (Pepple, 2013). Figure 1 represents the Openstack conceptual architecture with all native software components, developed by companies and individual supporters, depicting how they interact with each other.

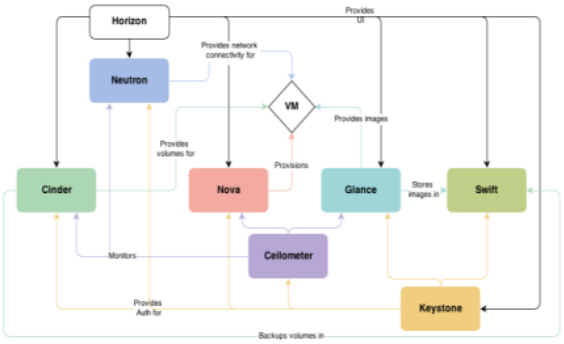

Figure 1 - Conceptual architecture (Openstack A, 2015)

Following, we describe those components dividing them into five groups due to their characteristics: Computing, Networking, Storing, Shared, and Supporting services. Although the supporting services are not native from Openstack's core, they are essential for its operation (Pepple, 2013).

2.1.1 Computing

The Nova Compute is the core software component of Openstack solution dictating services organization and interactions. Its own major components serve as basis for the cloud infrastructure, relying on Glance for Image services.

- *Nova*: Also known as Openstack Compute or Nova Compute, it provides virtual servers upon demand interacting with the hypervisors, supporting several kinds such as KVM, Xen, VMware or Hyper-V. It's an array of software that provides services for cloud resource management through its application programming interfaces (APIs), capable of orchestrating running instances, networks and access control. It's an essential service for a basic cloud architecture implementation.
- *Glance*: It's the Openstack Image service, a lookup and retrieval system for virtual machine (VM) images. It provides services for discovering, registering and retrieving virtual

images through an API that allows querying of VM image metadata, catalog and manage large libraries of server images. It's an essential service for a basic cloud architecture implementation.

2.1.2 Networking

Openstack has recently gained a new component for networking control and management, the Neutron.

- *Neutron*: It is a project to provide network connectivity as a service between devices managed by other Openstack services (e.g. Nova). It provides capabilities for managing dynamic host configuration protocol (DHCP), static protocols (IP's), or virtual networks (VLAN's) along with other advanced policies and topologies. Due to its architecture, it allows users to take advantage of frameworks (e.g. intrusion detection systems, load balancing, virtual private networks) from supported vendors.

2.1.3 Storing

When moving into a larger infrastructure for a wider service it could be an option to have dedicated servers for storage management. In this case Openstack offers the Swift and Cinder.

- *Swift*: Also known as Openstack Object Storage, Swift is a highly available, distributed object/blob store. Allows users to store or retrieve files. It can be used by Cinder component to back up the VMs volumes.
- *Cinder*: It is known as Openstack Block Storage or Cinder Volumes and provides persistent block storage (or volumes) to guest virtual machines. By working with Swift, Cinder can use it to backup the VMs volumes. In earlier releases of Openstack this service was an integral part of Nova in the form of nova-volume. Cinder will mainly interact with Nova, providing volumes for its instances, allowing through its API the manipulation of volumes, volume types and volume snapshots.

2.1.4 Shared Services

These are services transversal to all infrastructures, needed in order to put the solution in functioning. The available services are: Keystone, Horizon, and Ceilometer.

- *Keystone*: The Openstack identity is a single point of integration for Openstack policy, catalog, token and authentication, applying them to users and services interactions. Without Keystone there is no compliance between services consequently the cloud solutions will not be available for end users. It is an essential service for a basic cloud architecture implementation.
- *Horizon*: The Openstack dashboard is a modular web application that provides a user interface for cloud infrastructure management by interacting with the public APIs off all other services.
- *Ceilometer*: A new component that provides a configurable collection of metering data in terms of CPU and network costs available from all other services in the platform, delivering a unique point of contact for billing systems.

2.1.5 Supporting Services

The supporting services are crucial for Openstack to run, but are not part of the core components or developed for the solution, instead they are software from external vendors to provide internal support. Then, we briefly describe the Database and Advanced Message Queue Protocol (AMQP).

- *Database*: By default Openstack uses MySQL as its RDBMS for core services to store configurations and management information. Typically MySQL database is installed on the main node of the infrastructure, the controller node. It's an essential service for a basic cloud architecture implementation.
- *Advanced Message Queue Protocol*: AMQP is the messaging technology used in Openstack for inter-process communication. It uses by default RabbitMQ and supports others such as Qpid or ZeroMQ. The broker gives a common platform for processes to send and receive messages between them making use of little or no knowledge of each other's component definitions. It's an essential service for a basic cloud architecture implementation.

2.2 Service Architecture

The Nova Compute is the core software component of Openstack solution dictating services organization and interactions, as shown in Figure 2.

This component has as major components the API Server, Message Queue, Compute Worker or Compute Node, Network Controller, Volume Controller, Scheduler and Image Store.

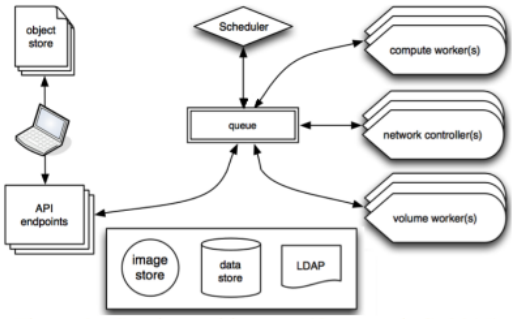

Figure 2 - Service architecture (Openstack C, 2015)

This API Server makes available to users the command and control of the hypervisor, storage and networking configurations through the API dashboard via basic http web services. A typical exchange of messages for an event request begins with the API server receiving user request, which authenticates and warrants that the user is permitted to do so. Is then evaluated the availability of resources requested by the user and, if available, the request is delivered to the queuing engine where services are listening to take action. When the work is done, a message is dispatched again into the queue for API server respond to the client.

2.3 Object Model

The object model represents the Openstack objects and how they stand before each other, showing how they relate, as depicted in Figure 3.

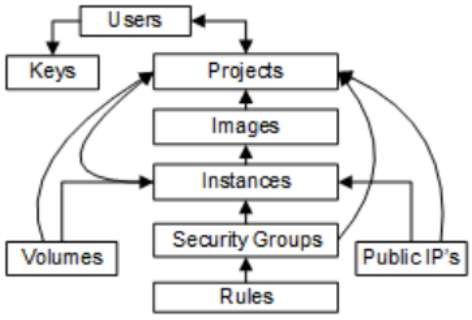

Figure 3 - Object model (Openstack B, 2015)

We have Users to access the cloud environment, Keys for authentication and security purposes and then Projects. Projects are private areas created inside cloud computing infrastructure by the administrator. We can have one single project with the total amount of resources of the infrastructure or create more than one configured individually for different purposes with different rules, policies, and virtual machines. Projects can be seen as smaller private clouds within enterprise's private cloud, a unique space tailored on demand for different departments with different needs giving them privacy and freedom without jeopardize other's work.

3 IMPLEMENTATION OF AN IAAS IN OPENSTACK

In this use case implementation we used Ubuntu Server 12.04 Long Term Support (LTS) for its reliability regarding longer support and maintenance and Havana has the last stable release of Openstack. In order to provide a clear insight that is feasible implement a robust, modular, scalable and yet still performant low cost cloud solution we used commercial range hardware.

3.1 Hardware Configuration

For easier understanding, Openstack uses the term Cloud Controller to identify the main node in the infrastructure and Compute Nodes for all the others. The technical main characteristics are the following:

- *Cloud controller*: Intel® Core™ i3-3220 CPU @ 3.30GHz, 8 GB DDR3 / 1067 MHz, 250Gb Hard Disc Drive. Added one extra Gigabit Ethernet controller.
- *Compute Node*: Intel® Core™ i3-3220 CPU @ 3.30GHz, 16 GB DDR3 / 1067 MHz, 250Gb Hard Disc Drive. Added one extra Gigabit Ethernet controller
- *Networking*: Two eight port Gigabit Ethernet switches

The network layout and the physical connections for the cloud implementation are described in Figure 4.

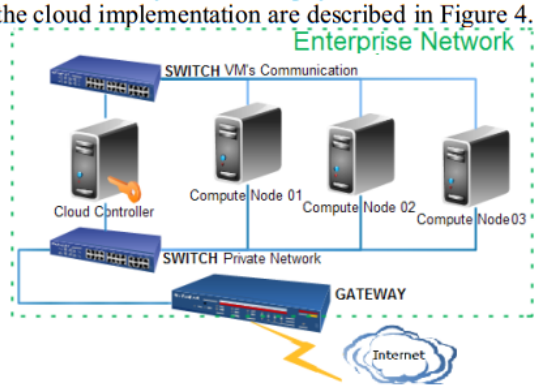

Figure 4 - Cloud network topology

All the nodes in the infrastructure communicate via two Gigabit private switches, one for VM's internal communications and another to ensure a connection to the enterprise internal network. The controller node is configured to do all the backend

processes via Nova, image management via Glance, hosting the website via Horizon and dealing with user identification via Keystone. All the Compute Nodes are configured with the Nova compute services and the KVM hypervisor.

Cloud architectures exploit virtualization techniques to provision multiple VM's on the same physical host so as to more efficiently use available resources.. With this basic configuration, built on entry level hardware, we aim to provide IT departments of companies' the guidelines for future implementations and exploitation of cloud computing technology based on Openstack modular architecture.

3.2 Software Configuration

Having chosen and implemented this basic architecture with legacy networking, in order to deliver an infrastructure as a service based on Openstack, there is no need to install all software components described for the architecture in order the cloud-computing environment to work.

Following we describe the software components to be installed on each machine, the main one who detains all the management activities for the environment, known as Cloud Controller, and remaining nodes or satellites named Compute Nodes, the machines that support all the available resources to be delivered.

3.2.1 Cloud Controller

The Cloud Controller is the core node in this environment. Following, we describe the services to be installed in order to provide such management capabilities that makes this the principal machine.

Nova is the core software component and is intended to be modular and easy to extend and adapt having the nova-api as a broker to its services in addition to native API Horizon. The nova-cert does all the certificate management for the platform for users and service's request flow. The nova-novncproxy is the software component that allows the access to instances through VNC clients bridging between public network and private network mediating token authentication and hypervisor-specific connections details. The connections are authorized by the validation of user's token provided by nova-consoleauth. The nova-scheduler manages users' requests for resources selecting the most suitable compute node for instances to run. The host selection criteria for dispatching requests considers factors like available physical memory, storage, processor or network load in a two-step process, which can be managed by the system administrator. The nodes in the system are filtered then weighted and sorted according to match those configured criteria. Glance is the image service for Openstack; it provides discovery, registration and delivery services for disk and server images, capable to copy or snapshot a server image to store it promptly or use them later as a template for new servers' creation. These capabilities are provided by glance-api, the focal point for image requests and glance-registry for processing images metadata like size, or type.

In this configuration, as in all others, shared services provided by horizon and keystone are needed in order to ease users' requests by being a single point of interaction between those requests and all the services responsible for delivering them and ensure identity among services, authentication and system policy, respectively.

Similar to Horizon and Keystone every Openstack implementation needs a database and a message queue. By default and here, we use MySQL for database management and RabbitMQ as the message broker for services. Figure 5 below, illustrates those services.

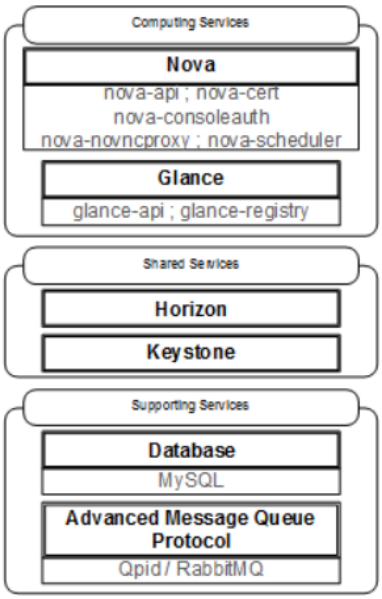

Figure 5 - Cloud Controller software modules

3.2.2 Compute Nodes

The compute nodes are the machines responsible for delivering to end users the resources available in the cloud platform under the form of virtual machines or storage. Although it has a layer of abstraction that allows him to run multiple hypervisors, it runs the

hypervisor portion of Compute using KVM by default as a virtualization infrastructure, which operates tenant virtual machines, as well as provisioning network services and implementing security groups using nova-compute and nova-network for that. Figure 6 shows the Compute Nodes software modules.

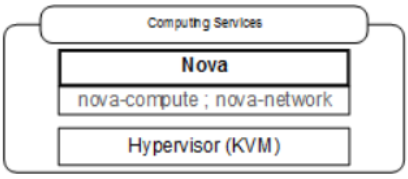

Figure 6 - Compute Nodes software modules

4 CONCLUSIONS

Cloud architectures exploit virtualization techniques to provision multiple Virtual Machines (VMs) on the same physical host, so as to efficiently use available resources. The cloud computing paradigm and available solutions in the market, regarding to Openstack, have been implemented and explored in various case scenarios, with major tests in academy and industry, presenting good results for performance, reliability and final user acceptance. Openstack leverages due to its modular architecture and simplicity to implement. It can be adopted into major datacentres but also to entry-level hardware providing enterprises a massively scalable cloud infrastructure. Allied to the fact of being open source, it helps on cost control, enables to deliver rapidly on-demand resources to final users.

This work has presented an overview of Openstack architecture, services available and how they work and adapt into any hardware infrastructure providing a sustainable and robust private cloud solution.

Based on this study, future work will involve an investigation on how to improve final user experience with the platform and administration activities regarding to measures and monitoring the cloud environment.

REFERENCES

Bist, M. Wariya, M. Agarwal, A. 2013. Comparing Delta, Open Stack and Xen Cloud Platforms: A Survey on Open Source IaaS. In 2013 3rd IEEE International Advance Computing Conference, 2013.

Castillo, J. A. L. Mallichan, K. Al-Hazmi, Y. 2013. Openstack federation in experimentation multi-cloud testbeds. In 2013 IEEE International Conference on Cloud Computing Technology and Science. 2013.

Chen, Y. Paxson, V. Katz, R. H. 2010. What's New About Cloud Computing Security? Electrical Engineering Computer Sciences, University of California at Berkeley, Technical Report No. UCB/EECS-2010-5. Jan. 2010.

Dargha, R. 2012. Cloud computing: from hype to reality. Fast tracking cloud adoption in International Conference on Advances. In computing, Communications and Informatics 2012, Aug 2012.

Eucalyptus – [online] https://www.eucalyptus.com (Accessed March 2015).

IaaS – [online] http://www.gartner.com/it-glossary/infrastructure-as-a-service-iaas (Accessed February 2015).

Kostanos, K. Kapsalis, A. Kyriazis, D. Themistocleous, M. Cunha, P. 2013. Open-source IaaS Fit for Purpose: A Comparison between Opennebula and Openstack. Int. Journ.. of Electronic Business Manag., Vol.11, Nº3, pp. 191-201, 2013.

Laszewski, G. Diaz, J. Wang, F. Fox, G. C. 2012. Comparison of multiple cloud frameworks. Pervasive Technology Institute, Indiana University, 2012.

OpenNebula – [online] http://opennebula.org/ (Accessed March 2015).

Openstack – [online] http://www.openstack.org/ (Accessed March 2015).

Openstack A. Conceptual architecture – [online] http://docs.openstack.org/juno/install-guide/install/apt/content/ch_overview.html (Accessed April 2015).

Openstack B. Object model – [online] http://docs.openstack.org/developer/nova/object.model.html (Accessed April 2015).

Openstack C. Service architecture – [online] http://docs.openstack.org/developer/nova/service.architecture.html (Accessed April 2015).

Pepple, K. 2013. Deploying Openstack, Creating opensource clouds, 2nd Edition, O'Reilly Media.

PaaS – [online] http://www.gartner.com/it-glossary/platform-as-a-service-paas/ (Accessed February 2015).

Sefraoui, O. Aissaoui, M. Eleuldj, M. 2012. OpenStack: Toward an Open-Source Solution for Cloud Computing. In International Journal of Computer Applications, Vol. 55 Nº3, October 2012.

SaaS – [online] http://www.techterms.com/definition/saas (Accessed February 2015).

Steinmetz, D. Perrault, B. Nordeen, R. Wilson, J. Wang, X. 2012. Cloud Computing Performance Benchmarking and Virtual Machine Launch Time. In SIGITE'12 Calgary, Alberta, Canada. 2012.

TaheriMonfared, A. Jaatun, M. G. 2011. As strong as the weakest link: Handling compromised components in OpenStack. In 3rd IEEE International Conference on Cloud Computing Technology and Science. 2011.

ANEXO D

Criação de imagens Windows com instalação de drivers virtIO

A criação das imagens *Windows* foi realizada numa máquina com sistema operativo *Linux* e com recursos à aplicação *Virtual Machine Manager 0.8.4* (*virt-manager*) para a criação das máquinas virtuais tendo sido baseado em (*Porting Windows to OpenStack, 2015*). Desta forma são criadas as imagens que final serão exportadas para que sejam carregadas na plataforma do *OpenStack*. Parte-se da premissa que o utilizador tem ao seu dispor uma imagem *ISO* ou um *DVD* do sistema operativo para uma instalação de raiz.

A versão atualizada dos drivers virtIO encontra-se disponível *online* em (*Windows VirtIO Drivers, 2014*).

Abrir a aplicação e criar uma máquina virtual atribuindo-lhe um nome (Figura D.1).

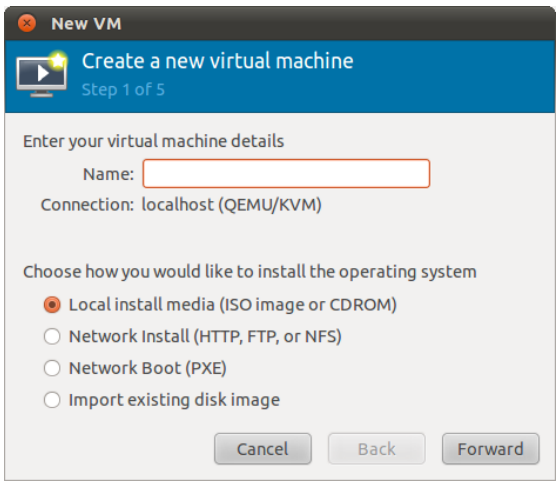

Figura D.1 - Criar nova imagem

Escolher a imagem ISO ou DVD com a imagem do sistema operativo e indicar qual a versão do Windows se vai utilizar (Figura D.2).

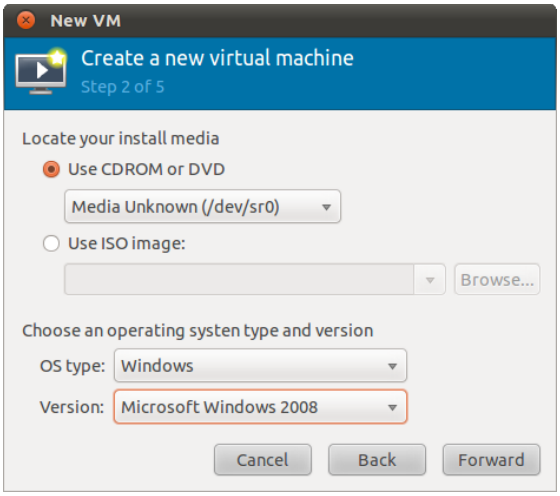

Figura D.2 - Selecionar a imagem do sistema operativo

Escolher os parâmetros de memória e CPU conforme exemplificado na Figura D.3. Estes parâmetros são livres, no sentido em que nada vão influenciar as opções a tomar futuramente na plataforma, servem unicamente para criar a máquina virtual que vai servir de suporte para criar a imagem do sistema operativo.

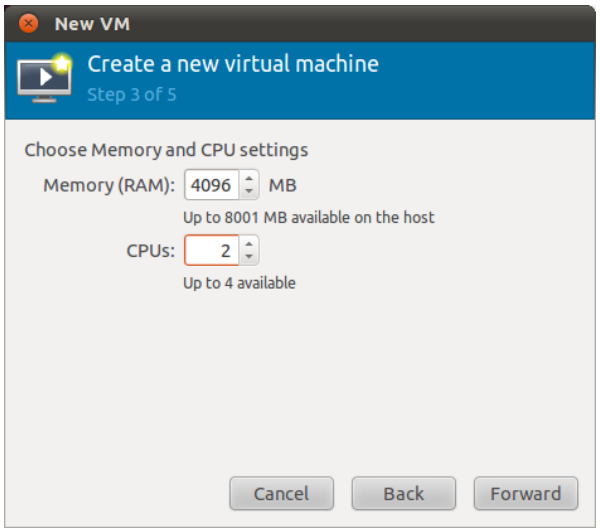

Figura D.3 - Definir parâmetros de memória e *CPU*

No quadro da Figura D.4 deve ser tirada a seleção para fazer a configuração do disco da VM. O mesmo vai ser feito mais à frente com a configuração dos drivers virtuais (*virtIO*).

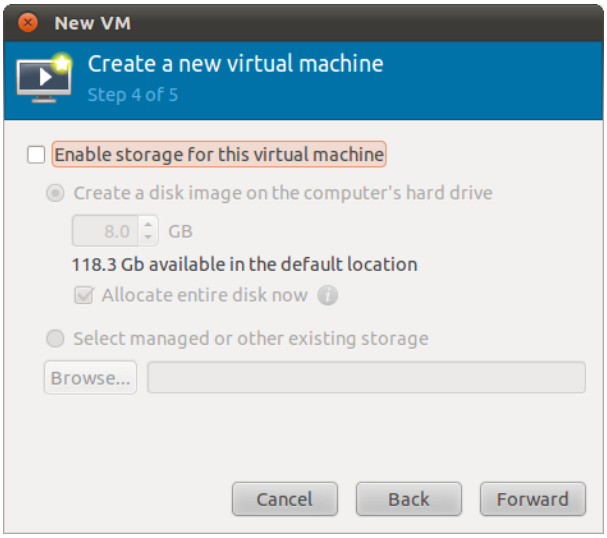

Figura D.4 - Desabilitar a configuração do disco

No quadro da Figura D.5 deve ser selecionada a opção de configuração personalizada.

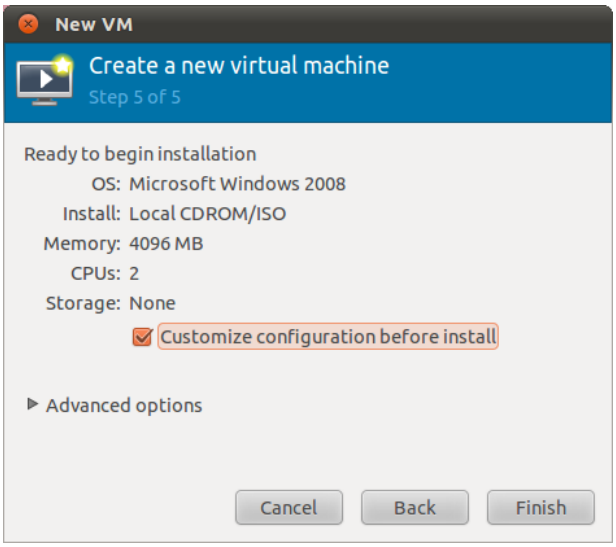

Figura D.5 - Selecionar a opção de configuração personalizada

Ao clicar em Finish no quadro da Figura D.5 vamos ser direcionados para o quadro de configurações de componentes conforme Figura D.6. Na listagem deve ser selecionada a placa de rede e configurado o modelo desta para *virtio* e clicar em aplicar. Depois, no fundo da janela, deve clicar em *Add Hardware*.

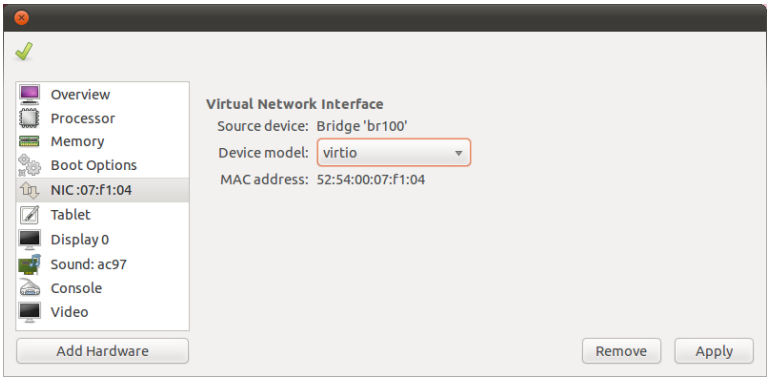

Figura D.6 - Configurar modelo da placa de rede

O novo *hardware* a adicionar é o disco rígido para a VM. Escolher o tipo *Storage* conforme exemplo da Figura D.7.

Figura D.7 - Adicionar unidade de disco rígido

Deve ser definido um tamanho para o disco e tirar a seleção de alocação imediata do tamanho total do disco conforme exemplo da Figura D.8.

Muito importante é escolher o tipo de disco como *Virtio Disk*.

Seguidamente clicar em *Forward* e no quadro seguinte que aparece, clicar em *Finish*.

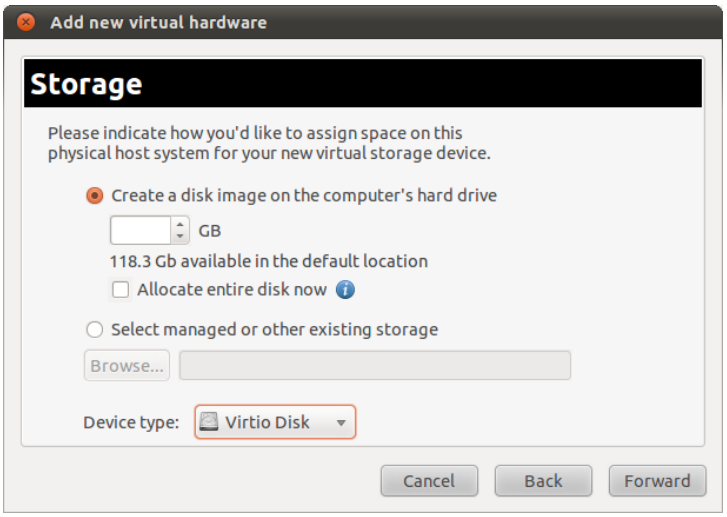

Figura D.8 - Configurar o disco rígido

Novamente no quadro das definições dos componentes, Figura D.9, deve selecionar o *CDROM* no menu *Boot Options* e clicar em *Apply* para guardar as alterações. Devemos agora mandar executar a máquina virtual (clicar no botão com o símbolo "*visto*" no canto superior esquerdo) para validar que não ocorre nenhum erro.

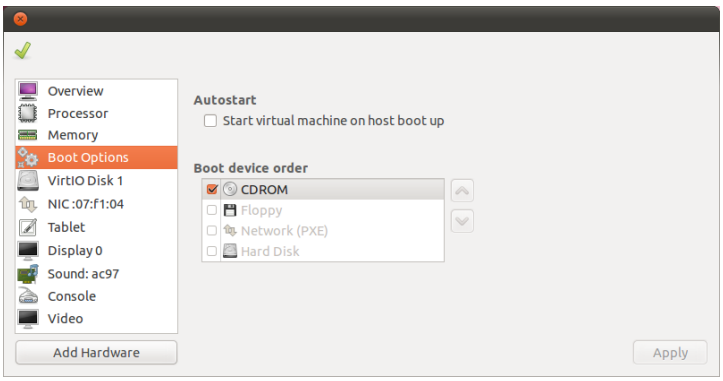

Figura D.9 - Selecionar a opção de *boot*

Tendo a máquina arrancada sem dar erro, devemos forçar a paragem das mesma, conforme Figura D.10, para configurar o disco com os drivers virtuais a utilizar mais á frente na instalação do sistema operativo.

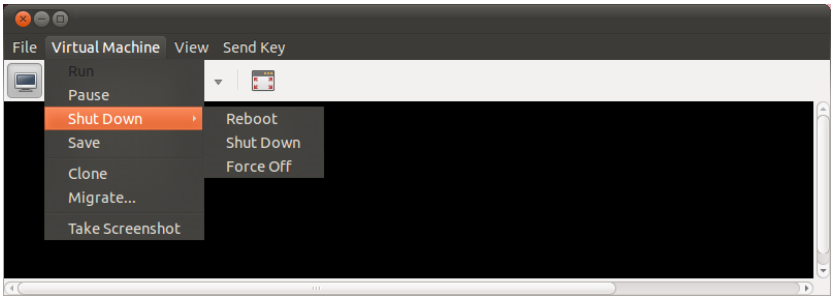

Figura D.10 - Forçar a aparagem da máquina virtual

Como a imagem faz sempre *boot* pelo *CDROM* com o número maior, deve ser reconfigurado o *CDROM* inicial para o *ISO* que tem os drivers virtuais conforme Figura D.11.

Seguidamente deve ser criado um novo *CDROM*, através do botão *Add Hardware* que se encontra no fundo da janela, e adicionar novamente o *ISO* ou o *DVD* com a imagem do sistema operativo.

Depois de reconfigurado os *CDROM's* devemos executar novamente o arranque da máquina virtual ao clicar no botão com a seta verde.

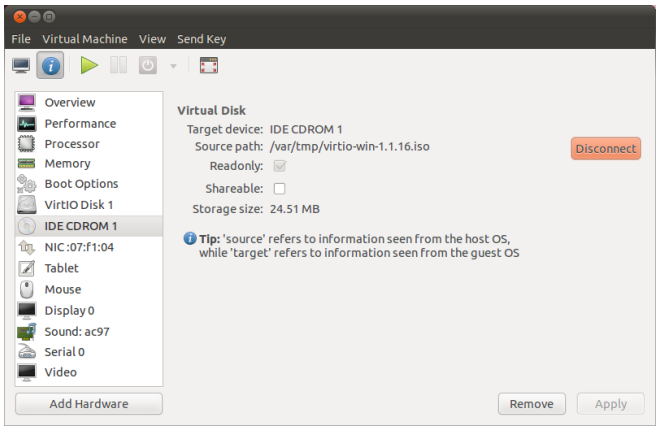

Figura D.11 - Configurar disco com os drivers *virtIO*

Neste momento aparece a imagem de uma comum instalação de sistema operativo conforme Figura D.12. Deve-se continuar com a instalação clicando em *Install now* e de seguida em *Custom (advanced)* para que seja possível selecionar o disco onde se vai instalar o sistema.

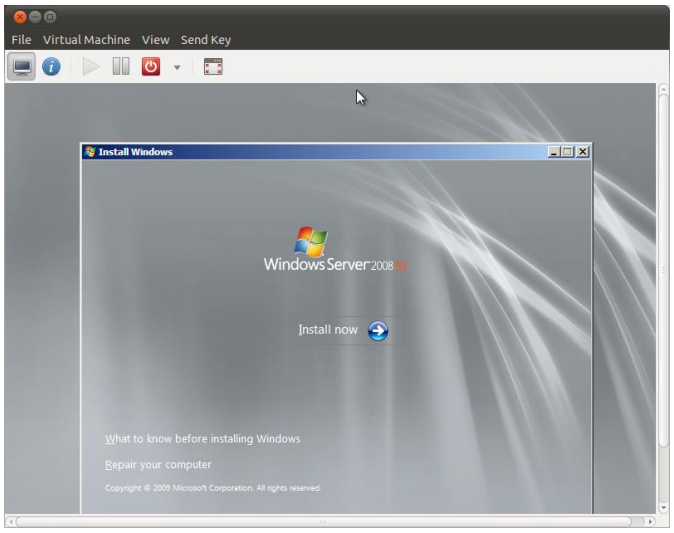

Figura D.12 - Instalação do sistema operativo

Neste momento a listagem dos discos aparece vazia devido a não existir ainda nenhum driver instalado para a gestão do disco virtual (Figura D.13), que foi o tipo de drive escolhida inicialmente. Para tal, devemos clicar em *Load Driver*.

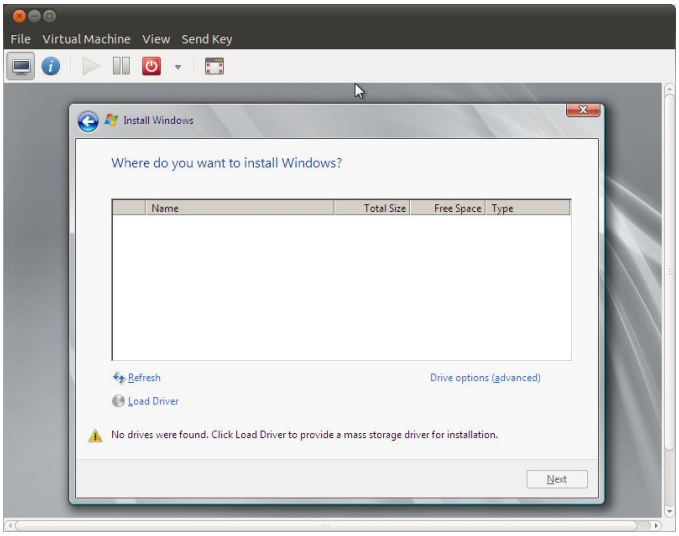

Figura D.13 - Instalação do sistema operativo (lista dos disco)

Na janela de *browser* que aparece representada na Figura D.14 devemos procurar o driver do disco que se encontra no CDROM que foi configurado no início da instalação. O driver a selecionar deverá ser o CDROM\viostor\Win7\amd64\viostor.inf

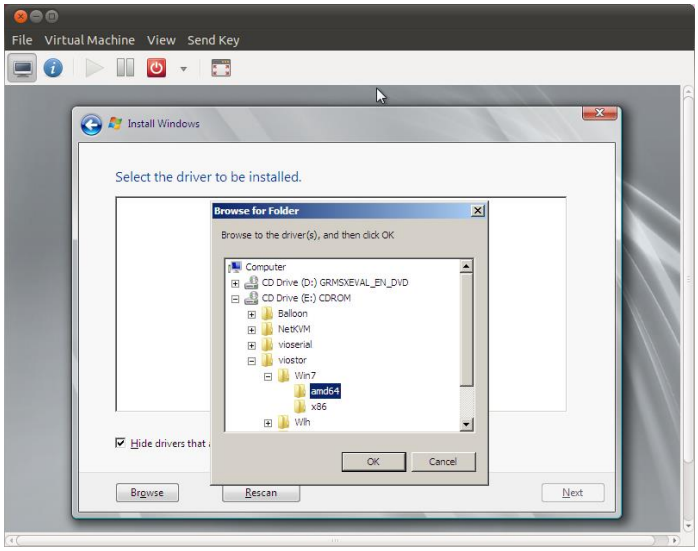

Figura D.14 - Selecionar drivers virtuais

Depois de selecionado o driver e clicar em *Next* para proceder à sua instalação, o disco para dar seguimento com a instalação do sistema operativo já irá aparecer na lista conforme representado na Figura D.15.

Ao selecionar o disco podemos continuar com a instalação do sistema operativo.

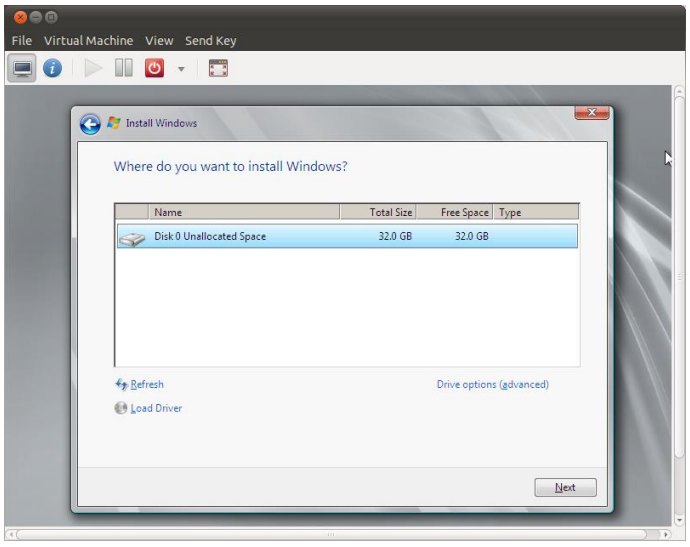

Figura D.15 - Disco para instalação do sistema operativo

No final da instalação do sistema operativo devemos fazer *shutdown* da máquina e alterar algumas das configurações, nomeadamente, remover o disco ou DVD com a imagem de instalação do sistema operativo e na opção *Boot Options* alterar de modo que só fique selecionado o *Hard Disk* conforme Figura D.16.

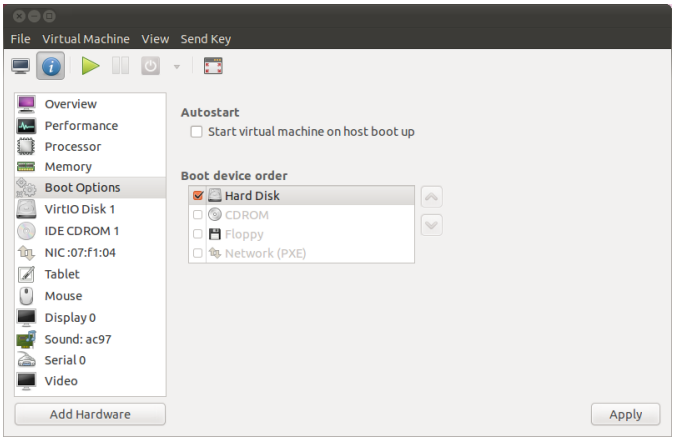

Figura D.16 - Alterar a ordem de *boot* da máquina virtual

Depois do primeiro login na máquina virtual é necessário aceder ao *Device Manager* para instalar os drivers em falta. Os drivers a utilizar para os dispositivos em falta devem ser pesquisados no CD com os drivers virtIO. No caso da placa de rede já aparecer instalada, devem desinstalar o driver e atualizar para o virtual.

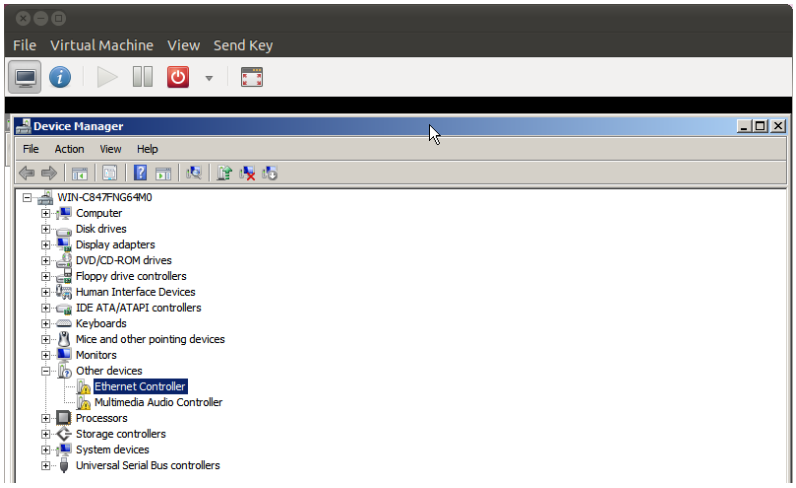

Figura D.17 - Instalação de *drivers* em falta após login

Para validar que tudo ficou a funcionar como esperado, deve ser reiniciada a máquina e feito novo *login*.

Todas as atualizações e instalação de *software* adicional deve ser feita agora.

Depois da imagem toda atualizada e personalizada de acordo com os objetivos, é necessário prepará-la para carregar no *OpenStack*. Para tal vamos recorrer ao *sysprep* do *Windows*, uma ferramenta que limpa toda a informação de segurança (*SID*) que torna a máquina única. Depois de executar o *sysprep* numa imagem, as máquinas virtuais criadas com base nela geram novos identificadores no momento do arranque, fazendo delas únicas.

Antes de executar o *sysprep* deve ser ativada a opção de *Windows Remote Desktop*.

O executável já se encontra nos ficheiros de sistema. O comando a executar deverá ser o seguinte: `C:\windows\System32\sysprep\sysprep.exe /generalize /oobe /shutdown`

Neste momento a imagem já se encontra pronta para ser carregada na plataforma do OpenStack.

ANEXO E

Instalação da ferramenta RALLY

A ferramenta utilizada para a realização do *benchamrk* apresentado no capítulo 7 desta dissertação recorreu à ferramenta *Rally*. A instalação e utilização da mesma é toda feita por linha de comandos. O presente anexo pretende demonstrar o processo de instalação e alguns comandos para a realização dos testes.

O pré-requisito para a instalação do *Rally* é a versão do *Python*.

```
# NOTE: The script assumes that you have the following
# programs already installed:
# -> Python 2.6, Python 2.7 or Python 3.4
```

Para validar a versão do *Python* que se encontra instalada na máquina executa-se o seguinte comando:

```
# python -V or python --version
```

O método mais simples para a instalação e posteriormente para ter disponíveis todos os *templates* com os testes, é criar uma cópia do repositório do *Github* na máquina onde for executar o *Rally*.

```
# git clone https://github.com/openstack/rally.git
/home/tiago/rally_repo
```

Depois de copiado o conteúdo do repositório é necessario instalar o Rally. Para tal existe um script para o efeito dentro do repositório acabado de copiar. Para tal devem solocar a linha de comando dentro dessa pasta e executar o script (com privilégios de administrador).

```
# sudo ./install_rally.sh
```

Após a instalação é necessário criar a base de dados do Rally.

```
# rally-manage db recreate
```

Em seguida é necessário registar na aplicação qual a plataforma de *Cloud* onde se pretende executar os testes de performance. Para tal, o método mais simples é criar um ficheiro **.json* com as configurações da plataforma. Conteúdo do ficheiro "*existingCloud.json*" que deverá ser preenchido com as configurações específicas de cada plataforma. Exemplo do ficheiro a criar:

```
{
    "type": "ExistingCloud",
    "auth_url": http://caminho_API":5000/v2.0/",
    "region_name": "RegionOne",
    "endpoint_type": "public",
    "admin": {
        "username": "admin",
        "password": "myadminpass",
```

```
            "tenant_name": "demo"
        },
        "https_insecure": False,
        "https_cacert": "",
    }
```

Seguidamente é necessário registar este ficheiro e por consequência a instalação de *Cloud* para que o *Rally* fique a saber onde executar os testes.

```
    # rally deployment create --file=existing.json --
name=existing
```

O método para validar que a configuração do ambiente foi bem-sucedida é executar o comando de validação do Rally.

```
    # rally deployment check
    keystone endpoints are valid and following services are
available:
    +----------+----------------+-----------+
    | services | type           | status    |
    +----------+----------------+-----------+
    | cinder   | volume         | Available |
    | glance   | image          | Available |
    | keystone | identity       | Available |
    | nova     | compute        | Available |
    +----------+----------------+-----------+
```

Ou então, como complemento, pedir listagens de algumas configurações da plataforma de Cloud, como a imagens disponíveis ou os flavors. Para tal, podem ser executados os seguintes comandos.

```
    # rally show images
    # rally show flavors
```

A partir deste momento já é possível executar os testes que se pretendam fazer. O teste é definido num ficheiro **.json* e executado da seguinte forma.

```
    # rally task start /home/tiago/Openstack/testes/boot-and-
delete.json
```

No final do teste executado é dada informação na linha de comandos de como fazer para gerar o relatório em *HTML* com os gráficos e os tempos do *benchmark* acabado de executar. O

código alfanumérico que aparece no exemplo que se segue é referente ao código único gerado para identificar cada teste.

```
  HINTS:
  * To plot HTML graphics with this data, run:
     rally task report 9c3831f5-96e5-415c-a4f0-add8cff24a70 --
out output.html
  * To generate a JUnit report, run:
     rally task report 9c3831f5-96e5-415c-a4f0-add8cff24a70 --
junit --out output.xml
  * To get raw JSON output of task results, run:
     rally task results 9c3831f5-96e5-415c-a4f0-add8cff24a70
```

ANEXO F

Ambiente gráfico do OpenStack

A página de login é bastante simples e direta. Apresenta o símbolo do OpenStack seguido do local para escrever as credenciais de acesso, conforme pode ser visto na Figura F.1

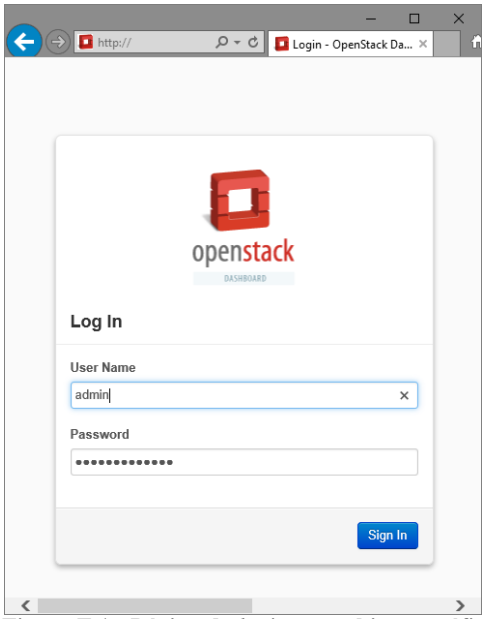

Figura F.1 - Página de *login* no ambiente gráfico

Após o utilizador fazer login na plataforma é direcionado para a vista geral do seu projeto, o qual lhe permite ter uma perceção dos recursos em utilização e quais ainda tem disponíveis, conforme se pode ver pelos gráficos circulares da Figura F.2. A azul está representado a ocupação face ao total disponível. Estes gráficos apresentam as instâncias, os *CPU's*, a memória *RAM*, os *IP's* públicos e os grupos de segurança.

Logo abaixo aparece uma listagem com as instâncias que se encontram em execução no momento, com um resumo das configurações de *CPU*, disco, memória e à quanto tempo estão em produção.

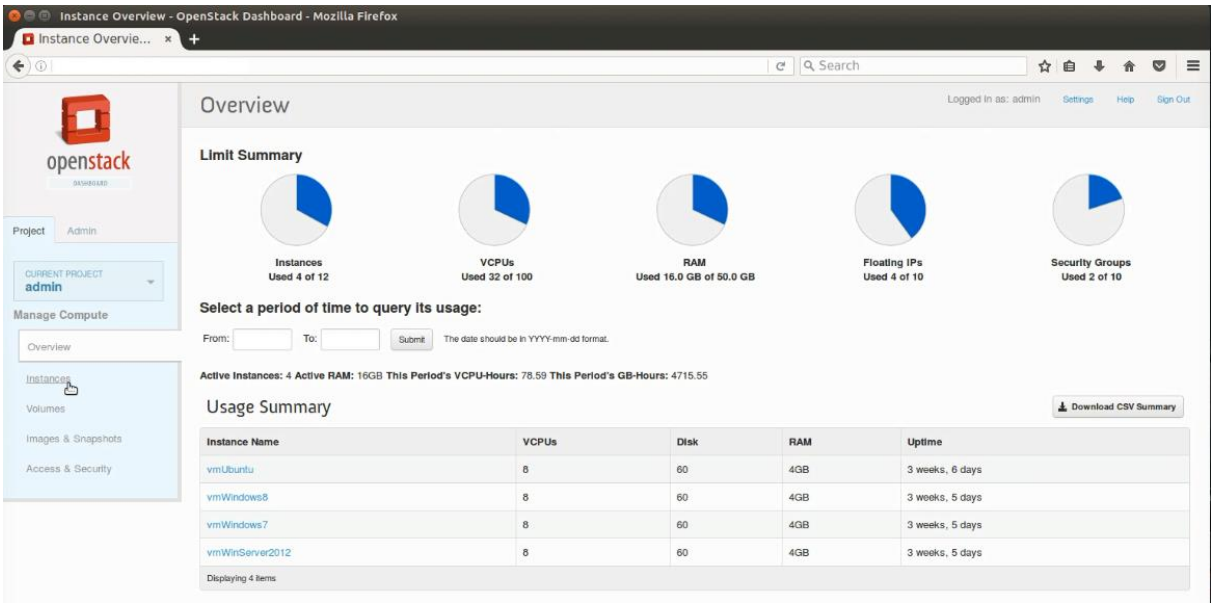

Figura F.2 - Página com a vista geral do projeto

No menu seguinte, *Instances*, apresentado na lateral esquerda do ambiente gráfico, estão as instancias que se encontram criadas nesse projeto, esteja elas em execução ou paradas. É nesta área, que se apresenta na Figura F.3, que se tomam as ações sobre as máquinas virtuais, como

por exemplo, a atribuição de um *IP* público, colocar em pausa, reiniciar, eliminar, redimensionar os parâmetros através da alteração do *flavor* ou criar uma imagem do estado atual da instância (*snapshot*).

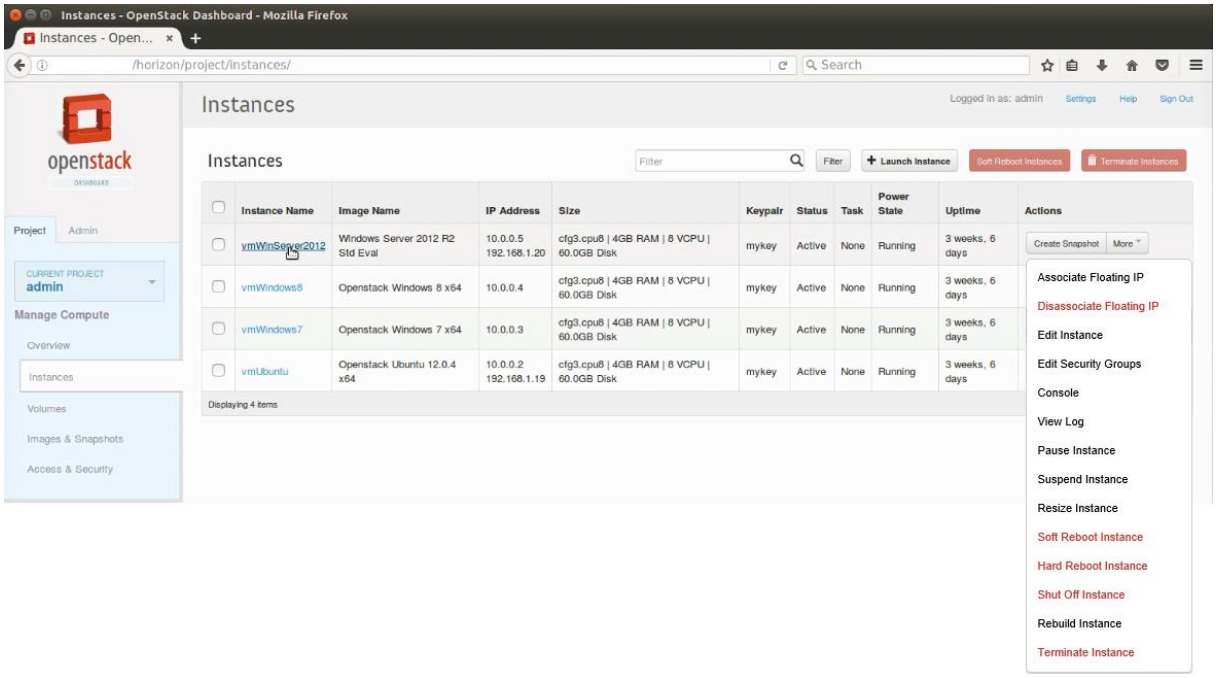

Figura F.3 - Listagem das instâncias do projeto

Ainda sobre a área de trabalho das instâncias, o nome de cada uma permite ao utilizador aceder á mesma a partir do *browser* de *Internet*, conforme se pode ver pela Figura F.4. Dependendo do tipo de tarefas que se pretendam executar sobre a máquina virtual, não é necessário atribuir-lhe um IP público para aceder a ela (que neste caso por publico entenda-se da gama de IP internos da empresa).

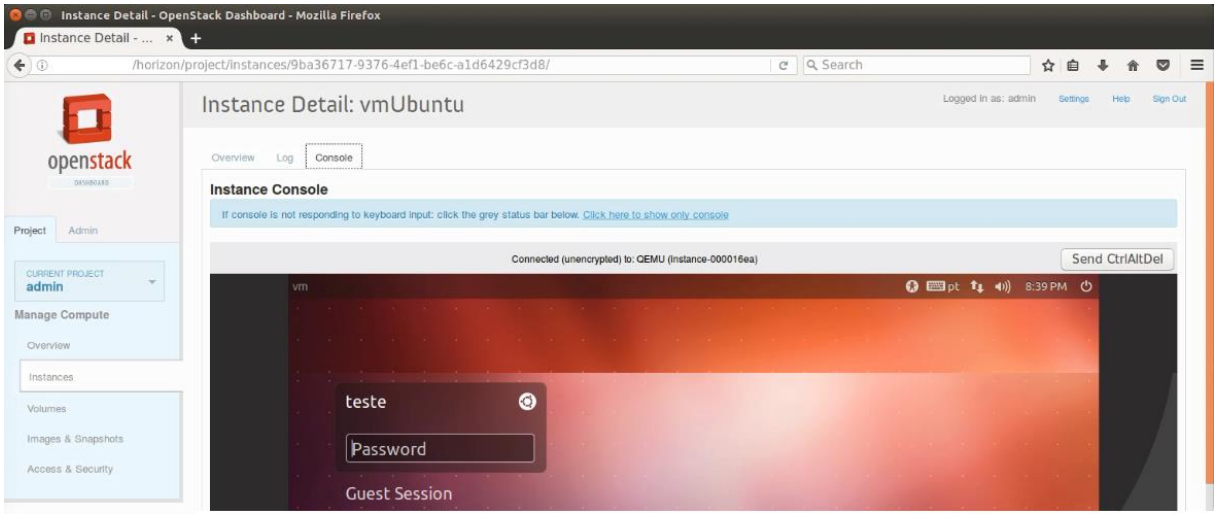

Figura F.4 - Acesso a máquina virtual diretamente a partir do *browser*

O menu seguinte, *Volumes*, que pode ser visto na Figura F.5, só é possível pois foi instalado o serviço *Cinder*. Este permite que sejam criados volumes, também conhecidos como discos virtuais, que por sua vez podem ser associados a uma das máquinas virtuais desse projeto.

No momento da criação de um projeto na plataforma, é definido a quantidade em *Gb* e o número de volumes que podem ser criados com esse mesmo espaço, a gestão do mesmo fica depois ao critério dos utilizadores.

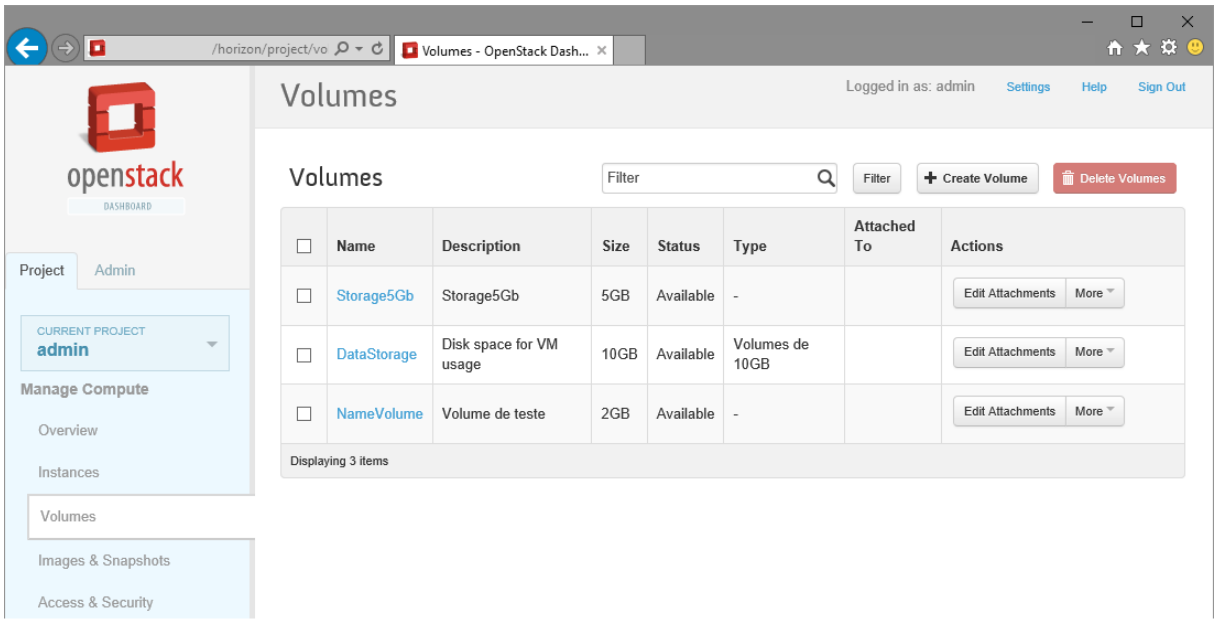

Figura F.5 - Gestão dos discos virtuais (*volumes*)

A Figura F.6 apresenta o menu *Images & Snapshots* onde o se encontra, como o nome sugere, a listagem das imagens e cópias de servidores. Estas imagens podem ser privadas, ou seja, afetas exclusivamente ao projeto onde foram criadas ou públicas, ficando dessa forma disponíveis para todos os projetos da plataforma.

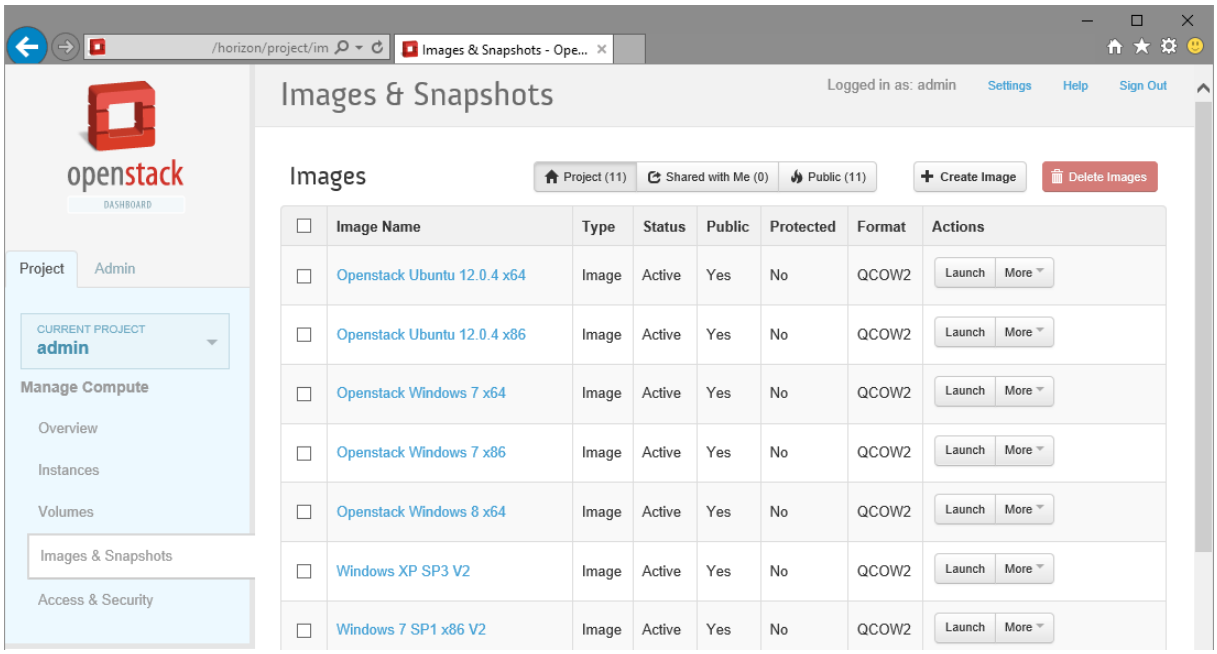

Figura F.6 - Imagens disponíveis para a criação de instâncias

Por ultimo, o menu *Access & Security*, apresentado na Figura F.7,permite definir regras de segurança que por sua vez podem ser aplicadas às máquinas virtuais. É neste mesmo local que estão definidos e é possível fazer a atribuição dos *IP's* públicos às máquinas virtuais.

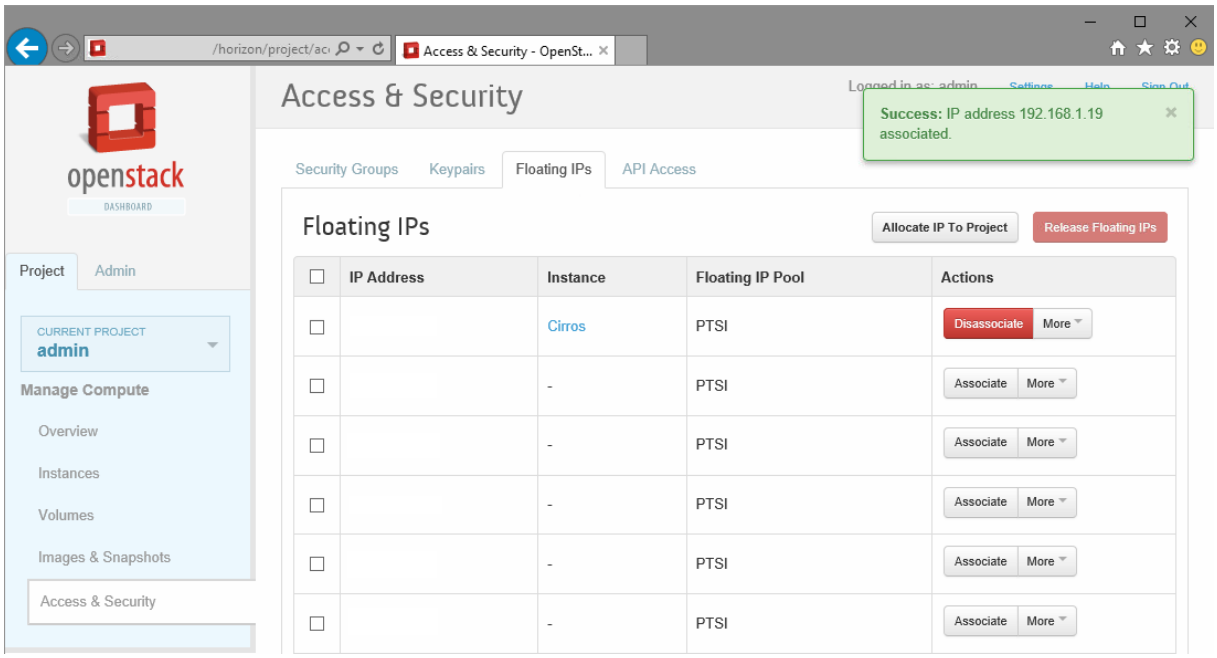

Figura F.7 - Definições de segurança

Quando no menu lateral se acede ao separador *Admin*, há alguns menus iguais aos que existem no projeto em que as funções deixam de estar restritas ao projeto em causa e passam a afetar todos os projetos. Mas para além destes, há menus que só se encontram disponíveis para o administrador.

Exemplo disso está representado na Figura F.8 onde o administrador tem à disposição os *flavors*, que são os *templates* utilizados para definir os parâmetros de computação (memória, disco e CPU) a atribuir às máquinas virtuais no momento da criação destas.

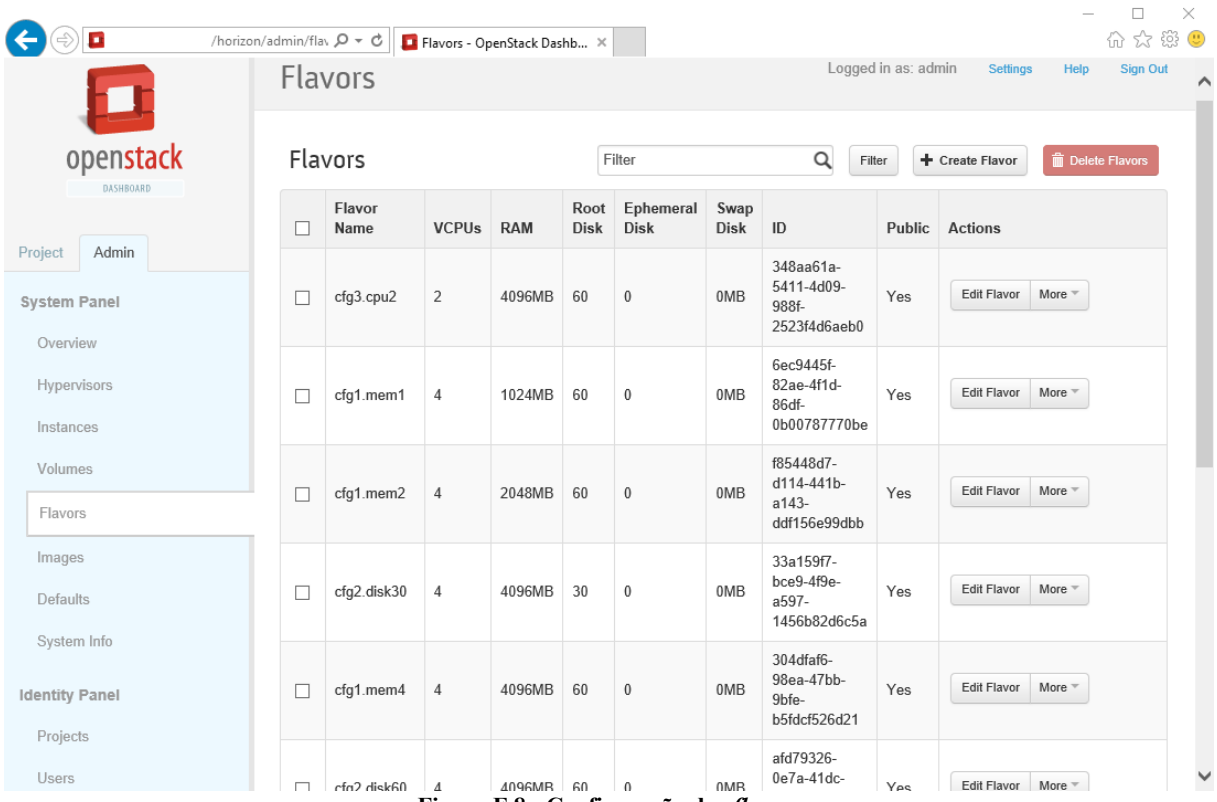

Figura F.8 - Configuração dos *flavors*

A página apresentada na Figura F.9 dá ao administrador informações sobre o sistema, mais concretamente sobre os serviços deste. No separador *Services* estão os serviços de base da plataforma (*Cinder*, *Glance*, *Nova*, *Keystone*) com a informação do servidor em que estão instalados e qual o estado, se ativos ou não. No separador *Compute Services* são apresentados os serviços de computação e de rede. Pode ver-se, por exemplo, que o serviço de rede só se encontra nos *Compute Nodes*, Host *openstack002*, *003* e *004*.

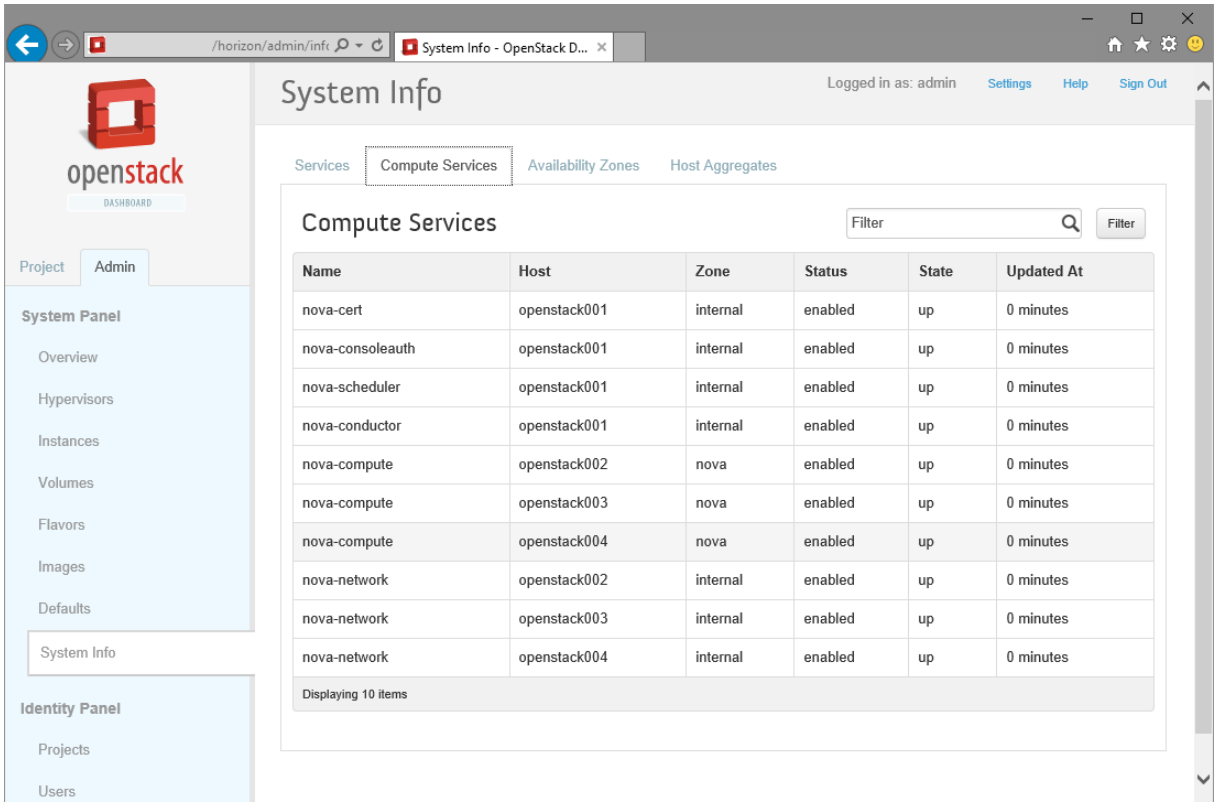

Figura F.9 - Informação do sistema

No separador *Defaults* estão definidos e é onde podem ser alterados os parâmetros de base para a criação dos projetos. Estão definidos valores como o número de instâncias, quantidade de memórias, *CPU's* ou quantidade de *IP's* públicos. Estes valores servem de base para a criação de um novo projeto, no entanto, podem ser modificados no momento da sua criação.

Por fim, ao fundo do menu lateral estão os menus para a administração dos utilizadores da plataforma e dos projetos.

ANEXO G

Tempos recolhidos para análise de performance

Tabela G.1 - Tempo de *boot* para amostra de 10 testes com variação de memória

Tempo de Boot para amostra de 10 testes														
SO	Memória	Disco	CPU	Concorrência	#01	#02	#03	#04	#05	#06	#07	#08	#09	#10
Ubuntu	1024	60	4	2	6,51	6,78	6,68	6,89	6,68	6,75	5,27	5,67	5,28	6,22
Win7	1024	60	4	2	5,06	6,82	6,02	5,11	5,02	4,96	5,55	4,99	6,03	7,14
Win8	1024	60	4	2	6,17	6,68	5,56	5,33	5,22	5,21	5,28	5,36	5,30	5,08
WS2012	1024	60	4	2	5,89	6,26	5,59	4,99	6,11	5,10	6,18	6,13	5,05	5,98
Ubuntu	2048	60	4	2	5,04	5,59	5,30	6,28	6,20	5,47	5,10	5,21	5,49	6,14
Win7	2048	60	4	2	7,08	7,43	6,69	5,48	5,46	6,36	5,29	5,29	5,25	6,17
Win8	2048	60	4	2	4,97	5,51	5,12	6,28	5,34	6,75	5,53	5,40	5,08	6,05
WS2012	2048	60	4	2	6,69	5,07	6,59	7,86	6,62	5,39	6,55	7,72	5,50	5,29
Ubuntu	4096	60	4	2	6,58	6,71	5,45	6,39	5,06	6,01	5,07	6,39	5,14	6,11
Win7	4096	60	4	2	5,13	5,79	5,00	5,02	6,04	6,02	5,04	6,60	7,64	8,84
Win8	4096	60	4	2	5,33	6,66	6,43	5,25	5,21	6,35	6,20	5,20	5,31	4,97
WS2012	4096	60	4	2	6,77	5,31	6,67	7,28	6,87	6,83	6,92	6,88	7,04	9,13
Ubuntu	1024	60	4	4	8,20	8,58	9,71	10,75	9,75	8,69	7,86	10,96	6,93	9,03
Win7	1024	60	4	4	7,85	8,72	8,98	10,38	9,40	9,53	7,97	10,75	6,92	5,35
Win8	1024	60	4	4	7,85	9,49	10,65	11,68	7,97	9,28	7,83	8,51	7,25	7,26
WS2012	1024	60	4	4	8,78	9,13	10,17	11,70	9,37	10,13	9,63	10,14	7,17	6,88
Ubuntu	2048	60	4	4	6,87	8,67	8,69	9,75	9,07	9,41	9,61	9,35	6,53	6,18
Win7	2048	60	4	4	8,65	9,41	9,96	11,02	8,17	9,20	7,16	11,62	6,95	9,12
Win8	2048	60	4	4	11,90	12,48	13,55	15,67	9,29	8,84	7,63	11,54	7,37	6,34
WS2012	2048	60	4	4	8,35	9,41	9,92	11,04	10,24	9,01	9,15	9,91	7,11	7,20
Ubuntu	4096	60	4	4	8,63	9,36	10,90	10,99	9,41	9,70	9,05	8,84	6,77	6,63
Win7	4096	60	4	4	9,32	9,48	10,39	10,89	10,22	10,01	10,00	10,57	7,00	6,92
Win8	4096	60	4	4	9,28	9,80	11,34	12,41	9,25	9,56	9,01	8,93	6,51	6,16
WS2012	4096	60	4	4	9,07	10,18	11,38	12,42	9,46	10,11	11,91	13,54	6,80	7,61
Ubuntu	1024	60	4	6	10,63	10,41	15,29	14,45	15,55	14,83	10,14	11,01	8,45	8,79
Win7	1024	60	4	6	6,83	10,84	14,24	15,32	16,46	16,61	9,07	11,05	8,82	8,25
Win8	1024	60	4	6	7,95	13,65	15,76	16,50	15,19	16,88	9,78	10,05	9,94	10,34
WS2012	1024	60	4	6	8,93	11,97	12,73	14,78	14,80	15,87	12,18	11,10	9,65	7,57
Ubuntu	2048	60	4	6	6,81	13,18	13,23	14,00	14,38	15,55	12,24	9,87	9,96	10,22
Win7	2048	60	4	6	12,59	15,57	16,64	16,64	15,61	17,85	11,34	10,04	9,40	9,64
Win8	2048	60	4	6	6,94	10,85	13,60	15,12	16,61	16,61	10,66	11,63	9,32	7,69
WS2012	2048	60	4	6	8,88	11,64	12,44	12,84	13,60	13,67	12,01	10,99	8,01	9,48
Ubuntu	4096	60	4	6	14,04	11,96	15,49	15,68	16,20	16,11	9,26	12,40	12,93	13,20
Win7	4096	60	4	6	9,41	12,93	15,13	14,32	14,50	17,32	11,50	11,46	12,29	10,93
Win8	4096	60	4	6	6,90	13,01	14,09	15,44	14,57	15,75	11,56	10,23	9,08	8,86
WS2012	4096	60	4	6	7,17	12,68	12,75	13,14	15,53	15,57	11,50	9,41	9,88	10,36

Tabela G.2 - Tempo de *delete* para amostra de 10 testes com variação de memória

Tempo de Delete para amostra de 10 testes														
SO	Memória	Disco	CPU	Concorrência	#01	#02	#03	#04	#05	#06	#07	#08	#09	#10
Ubuntu	1024	60	4	2	2,63	2,56	4,68	4,76	4,63	4,58	4,50	4,51	2,35	2,36
Win7	1024	60	4	2	4,43	4,41	4,37	4,47	2,37	4,85	4,58	4,75	2,37	4,48
Win8	1024	60	4	2	2,37	4,38	2,65	4,54	4,48	4,45	4,52	4,47	4,42	4,48
WS2012	1024	60	4	2	2,39	4,41	4,36	4,54	4,41	4,47	2,38	4,54	4,50	2,34
Ubuntu	2048	60	4	2	4,38	4,45	4,52	4,46	4,38	4,55	2,35	4,38	2,38	4,51
Win7	2048	60	4	2	2,39	4,44	4,38	4,39	4,52	4,49	2,42	4,47	4,47	2,37
Win8	2048	60	4	2	4,39	4,81	4,60	4,59	4,43	4,40	4,51	4,41	4,42	4,41
WS2012	2048	60	4	2	2,51	4,45	4,44	4,53	4,39	4,47	4,39	4,41	4,42	4,46
Ubuntu	4096	60	4	2	2,50	4,56	4,37	4,41	4,65	4,39	4,48	4,59	4,49	2,33
Win7	4096	60	4	2	4,39	4,52	4,41	4,43	2,34	4,94	2,44	4,41	4,41	4,41
Win8	4096	60	4	2	4,40	4,69	4,44	4,45	4,34	4,53	4,41	4,57	4,41	4,45
WS2012	4096	60	4	2	2,34	4,40	4,82	4,87	4,63	4,75	4,67	4,69	4,58	4,57
Ubuntu	1024	60	4	4	4,90	4,63	4,56	4,78	5,19	5,22	4,99	4,51	4,66	4,62
Win7	1024	60	4	4	4,44	4,79	4,62	4,95	4,60	4,85	4,61	4,60	4,67	4,68
Win8	1024	60	4	4	4,83	4,53	4,57	4,65	4,94	4,96	4,45	4,50	4,70	4,71
WS2012	1024	60	4	4	4,62	4,57	4,64	4,77	4,59	4,45	4,79	4,88	4,73	4,75
Ubuntu	2048	60	4	4	4,69	5,13	5,12	4,86	4,63	3,09	5,13	4,60	4,38	4,41
Win7	2048	60	4	4	4,57	4,80	4,49	4,46	5,60	5,59	5,47	4,48	4,51	4,48
Win8	2048	60	4	4	5,18	4,71	4,79	4,91	4,52	4,61	4,50	4,49	2,35	4,39
WS2012	2048	60	4	4	4,49	4,66	4,42	4,65	5,04	5,12	4,76	4,43	4,68	4,55
Ubuntu	4096	60	4	4	4,66	4,65	4,90	5,22	4,44	4,62	4,84	4,82	4,44	4,43
Win7	4096	60	4	4	4,67	4,72	4,42	5,15	4,79	4,87	4,59	4,48	4,77	4,77
Win8	4096	60	4	4	4,64	4,56	4,78	4,64	4,41	4,70	4,38	4,79	2,32	2,39
WS2012	4096	60	4	4	4,56	4,66	4,55	4,77	6,73	8,59	4,65	4,70	2,45	2,42
Ubuntu	1024	60	4	6	4,76	6,91	5,52	6,85	7,35	9,11	4,37	6,48	4,55	4,42
Win7	1024	60	4	6	6,71	4,65	6,83	6,76	7,64	7,63	4,98	4,99	4,61	4,55
Win8	1024	60	4	6	6,64	4,70	4,86	4,53	7,90	6,54	4,49	4,57	4,81	6,86
WS2012	1024	60	4	6	4,70	4,83	6,58	7,61	7,61	6,62	4,51	4,63	4,45	4,47
Ubuntu	2048	60	4	6	6,72	5,18	5,22	5,14	7,19	6,83	4,76	4,75	4,61	4,40
Win7	2048	60	4	6	5,27	4,83	5,69	5,83	6,86	6,92	4,53	4,47	4,47	4,41
Win8	2048	60	4	6	6,85	4,48	5,11	6,83	6,95	6,98	4,64	4,51	4,46	4,61
WS2012	2048	60	4	6	4,58	4,80	7,17	6,81	7,29	7,23	4,55	4,81	4,84	4,50
Ubuntu	4096	60	4	6	4,70	6,83	5,30	5,47	5,09	7,34	2,54	4,39	4,39	4,41
Win7	4096	60	4	6	6,67	4,58	4,79	6,97	6,92	6,97	4,61	4,41	4,74	4,78
Win8	4096	60	4	6	6,51	4,70	4,99	4,94	7,18	6,70	4,63	4,94	4,76	4,50
WS2012	4096	60	4	6	4,78	5,24	5,26	5,39	7,09	7,09	4,72	4,53	4,57	4,62

Tabela G.3 - Tempo de *boot* para amostra de 10 testes com variação de disco

Tempo de Boot para amostra de 10 testes														
SO	Memória	Disco	CPU	Concorrência	#01	#02	#03	#04	#05	#06	#07	#08	#09	#10
Ubuntu	4096	30	4	2	5,35	6,94	6,32	5,22	5,87	5,46	6,18	7,53	6,57	7,74
Win7	4096	30	4	2	6,14	6,63	5,59	6,29	5,45	5,39	6,48	5,25	5,15	5,15
Win8	4096	30	4	2	5,03	5,46	5,49	6,69	5,43	6,10	6,14	5,31	5,41	6,74
WS2012	4096	30	4	2	5,11	5,68	5,22	4,89	5,26	5,16	5,49	5,35	6,55	6,22
Ubuntu	4096	60	4	2	5,99	6,47	5,45	6,52	6,26	7,47	6,39	7,57	5,31	5,35
Win7	4096	60	4	2	6,66	6,81	6,95	6,86	5,45	6,92	5,16	5,23	6,16	6,12
Win8	4096	60	4	2	6,56	6,74	5,73	5,29	6,28	5,46	5,42	5,49	5,52	6,49
WS2012	4096	60	4	2	6,79	6,29	6,22	6,22	5,23	6,20	5,17	5,32	6,57	6,24
Ubuntu	4096	120	4	2	6,99	6,67	6,45	7,52	6,56	7,47	7,39	7,57	6,31	6,35
Win7	4096	120	4	2	7,66	7,81	8,95	6,16	6,45	6,98	6,16	5,75	7,16	6,76
Win8	4096	120	4	2	7,13	8,74	6,73	6,29	7,05	6,46	6,42	6,21	6,52	7,49
WS2012	4096	120	4	2	7,79	7,29	6,52	6,72	8,23	6,27	7,17	6,32	6,99	7,24
Ubuntu	4096	30	4	4	9,36	10,28	10,29	12,37	7,95	7,49	9,87	9,76	6,13	7,45
Win7	4096	30	4	4	8,38	9,00	9,23	11,69	9,20	9,14	9,26	9,79	6,52	6,57
Win8	4096	30	4	4	8,07	8,24	9,65	10,73	9,96	10,01	9,57	9,72	5,46	7,07
WS2012	4096	30	4	4	8,31	8,68	9,75	10,83	8,19	10,55	10,59	11,65	6,29	6,38
Ubuntu	4096	60	4	4	8,80	9,21	9,87	11,29	9,83	10,37	10,32	12,37	6,49	6,58
Win7	4096	60	4	4	9,03	9,05	9,66	10,45	9,24	9,63	10,47	9,76	6,49	7,52
Win8	4096	60	4	4	8,67	10,59	10,82	10,83	9,51	10,31	9,52	10,17	6,78	6,57
WS2012	4096	60	4	4	10,90	10,93	12,08	11,74	8,35	9,93	9,92	9,18	6,75	6,58
Ubuntu	4096	120	4	4	8,12	9,64	10,39	10,46	8,97	11,27	11,37	11,90	8,03	7,74
Win7	4096	120	4	4	8,62	8,73	9,58	12,04	8,76	10,61	10,76	8,79	7,93	6,81
Win8	4096	120	4	4	8,41	9,86	11,28	12,35	8,65	8,55	9,95	9,39	9,08	8,15
WS2012	4096	120	4	4	9,30	10,21	11,28	12,28	10,62	8,95	9,87	8,56	6,39	6,49
Ubuntu	4096	30	4	6	6,59	9,55	13,47	14,39	15,62	16,04	11,70	9,84	9,24	7,46
Win7	4096	30	4	6	8,74	11,71	12,26	13,99	13,99	14,96	13,65	10,34	9,61	11,24
Win8	4096	30	4	6	7,21	10,05	14,20	14,38	13,37	15,86	10,41	11,54	9,08	10,70
WS2012	4096	30	4	6	11,85	12,99	15,49	15,70	16,25	17,10	10,51	9,33	10,34	11,19
Ubuntu	4096	60	4	6	8,27	14,81	13,01	15,26	16,46	15,27	12,25	10,25	10,57	11,31
Win7	4096	60	4	6	6,47	11,35	14,01	14,16	15,45	16,56	10,44	10,54	7,44	7,95
Win8	4096	60	4	6	7,69	12,60	14,28	15,37	14,12	16,47	12,73	8,60	10,25	9,08
WS2012	4096	60	4	6	11,14	14,87	15,32	15,31	18,64	18,61	10,89	8,91	10,99	12,00
Ubuntu	4096	120	4	6	9,68	14,48	15,54	15,50	18,01	18,26	10,78	8,70	10,36	10,00
Win7	4096	120	4	6	7,86	11,04	17,59	15,95	16,49	17,77	10,71	11,41	9,77	11,64
Win8	4096	120	4	6	5,76	14,69	17,06	15,65	16,09	16,11	8,07	10,69	7,96	10,61
WS2012	4096	120	4	6	9,14	12,41	12,74	14,85	14,92	14,33	11,45	11,61	11,59	9,00

Tabela G.4 - Tempo de *delete* para amostra de 10 testes com variação de disco

Tempo de Delete para amostra de 10 testes														
SO	Memória	Disco	CPU	Concorrência	#01	#02	#03	#04	#05	#06	#07	#08	#09	#10
Ubuntu	4096	30	4	2	4,42	4,51	2,47	4,38	4,44	2,42	4,44	4,39	4,43	2,43
Win7	4096	30	4	2	2,40	4,41	2,46	4,75	4,48	4,53	4,62	4,43	4,49	2,47
Win8	4096	30	4	2	4,44	4,41	4,46	4,52	4,46	4,43	2,37	4,38	4,42	2,36
WS2012	4096	30	4	2	4,41	4,85	4,49	4,41	4,41	4,41	2,46	4,41	2,47	4,39
Ubuntu	4096	60	4	2	2,38	2,45	5,15	5,06	4,71	4,38	4,69	4,65	2,46	2,40
Win7	4096	60	4	2	4,61	4,61	4,66	4,63	4,36	4,51	4,40	4,43	4,46	4,45
Win8	4096	60	4	2	2,53	4,67	4,43	4,42	2,38	4,45	4,48	2,44	2,47	4,54
WS2012	4096	60	4	2	2,36	4,53	2,42	4,45	4,46	4,59	4,39	2,56	4,47	4,44
Ubuntu	4096	120	4	2	4,63	4,85	4,73	4,57	2,72	4,73	4,43	6,82	2,35	4,45
Win7	4096	120	4	2	4,93	4,94	4,65	4,61	4,68	4,73	4,49	4,60	4,40	2,42
Win8	4096	120	4	2	4,44	5,25	4,38	4,81	4,79	4,61	7,01	4,48	4,73	2,32
WS2012	4096	120	4	2	4,45	4,86	4,88	4,86	5,14	4,95	4,99	4,67	4,66	4,67
Ubuntu	4096	30	4	4	4,38	4,75	4,79	4,67	4,47	2,70	5,31	7,38	3,05	2,92
Win7	4096	30	4	4	4,60	4,74	4,94	4,63	4,54	4,52	6,55	4,59	2,37	4,38
Win8	4096	30	4	4	5,02	4,88	4,38	4,80	4,98	5,01	4,50	4,66	2,33	4,47
WS2012	4096	30	4	4	4,54	4,52	4,57	4,56	4,70	4,45	6,62	4,56	4,48	2,43
Ubuntu	4096	60	4	4	4,49	4,52	4,62	5,28	2,80	6,85	5,13	7,14	4,50	2,37
Win7	4096	60	4	4	4,79	4,93	4,50	4,42	4,76	4,58	4,94	4,95	4,51	4,39
Win8	4096	60	4	4	4,59	3,00	5,00	5,11	4,57	4,71	4,42	4,84	4,38	4,53
WS2012	4096	60	4	4	4,69	4,68	4,69	6,74	2,86	4,72	4,82	4,58	2,46	4,49
Ubuntu	4096	120	4	4	4,71	4,73	5,62	5,55	4,55	6,79	6,84	4,77	4,72	4,43
Win7	4096	120	4	4	5,36	5,34	4,57	4,88	4,60	4,83	4,79	4,54	4,46	4,48
Win8	4096	120	4	4	4,64	4,51	4,45	4,72	4,71	5,37	5,35	6,94	4,53	4,49
WS2012	4096	120	4	4	4,40	4,67	4,38	5,04	4,51	4,77	4,68	4,67	2,37	4,41
Ubuntu	4096	30	4	6	6,64	6,94	4,67	7,23	6,98	6,76	4,66	4,41	4,47	4,53
Win7	4096	30	4	6	4,82	4,75	4,71	5,15	5,18	5,91	5,19	5,17	4,48	6,86
Win8	4096	30	4	6	6,90	4,71	4,97	4,84	7,36	7,17	4,49	4,60	4,55	4,48
WS2012	4096	30	4	6	6,76	6,84	4,91	5,15	5,00	5,04	4,44	4,75	4,78	4,49
Ubuntu	4096	60	4	6	6,80	5,23	7,05	5,58	4,84	7,57	4,46	4,59	4,45	4,45
Win7	4096	60	4	6	6,50	4,72	7,22	7,14	6,65	6,95	4,66	4,69	4,56	4,49
Win8	4096	60	4	6	7,02	4,85	4,82	5,15	6,97	6,70	4,87	5,08	5,09	4,49
WS2012	4096	60	4	6	6,81	5,56	7,20	7,28	7,38	7,42	4,60	4,51	4,46	4,36
Ubuntu	4096	120	4	6	6,61	6,65	6,95	7,01	7,00	6,81	6,76	4,82	4,70	4,48
Win7	4096	120	4	6	6,65	4,91	5,19	6,98	6,81	7,78	5,16	4,45	4,49	4,55
Win8	4096	120	4	6	6,82	4,65	4,57	7,39	7,37	7,35	6,91	4,56	4,49	4,45
WS2012	4096	120	4	6	4,91	4,99	4,69	6,07	6,18	6,92	4,69	4,97	4,98	4,48

Tabela G.5 - Tempo de *boot* para amostra de 10 testes com variação de CPU

Tempo de Boot para amostra de 10 testes														
SO	Memória	Disco	CPU	Concorrência	#01	#02	#03	#04	#05	#06	#07	#08	#09	#10
Ubuntu	4096	60	2	2	5,52	4,98	6,80	6,02	5,34	6,35	5,16	5,19	6,26	5,37
Win7	4096	60	2	2	5,74	7,51	6,19	5,27	5,52	5,12	5,24	6,16	6,14	6,16
Win8	4096	60	2	2	5,12	6,70	6,35	6,42	5,69	6,43	6,44	5,36	6,57	6,36
Ws2012	4096	60	2	2	6,06	6,53	6,49	6,48	5,49	6,49	5,45	5,85	5,51	5,29
Ubuntu	4096	60	4	2	6,81	6,20	6,44	6,04	5,19	6,17	5,05	5,37	6,36	5,13
Win7	4096	60	4	2	6,93	5,43	6,12	6,28	6,36	6,12	6,20	6,11	6,22	5,33
Win8	4096	60	4	2	6,08	6,93	6,17	6,21	5,38	6,38	6,63	6,75	5,08	5,36
Ws2012	4096	60	4	2	6,15	6,59	6,65	5,28	5,73	5,90	5,19	5,11	6,49	5,45
Ubuntu	4096	60	8	2	6,32	6,84	6,46	6,40	5,15	6,50	6,12	5,38	5,35	5,36
Win7	4096	60	8	2	6,62	5,14	7,15	7,13	7,02	7,13	7,28	5,92	5,24	4,96
Win8	4096	60	8	2	6,31	6,78	5,26	6,32	5,80	6,37	6,19	5,31	6,35	6,33
Ws2012	4096	60	8	2	6,34	8,94	6,48	6,25	5,43	6,40	6,31	5,21	5,28	5,41
Ubuntu	4096	60	2	4	7,89	8,84	9,23	10,74	7,55	9,05	8,71	9,82	6,67	7,23
Win7	4096	60	2	4	8,40	8,89	9,36	10,88	9,36	9,13	8,64	9,37	7,17	8,86
Win8	4096	60	2	4	8,43	9,38	10,51	11,57	9,24	8,87	8,00	8,46	6,23	6,19
Ws2012	4096	60	2	4	7,72	8,70	8,72	11,09	8,11	7,93	8,87	7,98	6,46	7,58
Ubuntu	4096	60	4	4	8,17	9,47	9,98	10,52	10,20	9,25	8,92	10,62	7,20	6,18
Win7	4096	60	4	4	8,14	8,66	10,72	10,74	10,00	10,10	8,68	9,23	7,76	6,91
Win8	4096	60	4	4	7,96	8,58	9,95	10,74	8,84	7,81	10,88	12,39	7,71	7,68
Ws2012	4096	60	4	4	7,80	9,22	9,27	10,37	8,05	9,37	9,82	11,21	7,26	6,63
Ubuntu	4096	60	8	4	9,21	9,48	9,73	11,37	9,98	10,78	10,46	9,89	6,39	7,63
Win7	4096	60	8	4	8,41	9,03	9,07	11,16	8,52	10,48	13,14	14,59	7,85	6,70
Win8	4096	60	8	4	8,64	8,91	9,98	11,05	9,43	9,73	9,35	9,98	5,28	6,56
Ws2012	4096	60	8	4	8,52	8,50	9,13	10,89	7,52	8,62	8,91	8,68	6,49	5,36
Ubuntu	4096	60	2	6	7,85	13,24	13,59	14,61	15,12	15,12	9,13	13,33	9,65	11,74
Win7	4096	60	2	6	7,53	13,29	13,95	14,02	15,61	15,62	10,99	8,73	9,93	10,52
Win8	4096	60	2	6	8,42	11,78	14,45	14,37	15,06	15,12	10,92	11,50	8,09	9,04
Ws2012	4096	60	2	6	7,77	10,37	12,22	13,60	13,59	14,71	11,18	9,13	11,23	8,22
Ubuntu	4096	60	4	6	10,30	11,61	13,60	13,58	14,41	14,41	11,62	11,48	8,96	9,99
Win7	4096	60	4	6	7,99	12,28	15,16	14,82	19,25	18,19	10,60	9,62	11,22	10,81
Win8	4096	60	4	6	8,88	12,79	13,39	13,91	18,03	18,01	10,54	12,33	14,62	14,37
Ws2012	4096	60	4	6	8,49	13,19	16,04	16,14	16,81	17,29	10,24	10,93	10,17	10,28
Ubuntu	4096	60	8	6	7,37	10,81	14,93	15,13	15,70	19,12	8,94	14,23	14,51	11,87
Win7	4096	60	8	6	15,61	15,61	16,62	17,52	15,93	16,62	9,64	10,46	9,41	11,48
Win8	4096	60	8	6	5,95	16,32	15,03	15,53	17,65	5,57	18,46	7,98	10,61	10,82
Ws2012	4096	60	8	6	8,41	13,14	13,67	15,31	15,64	16,92	12,18	10,85	11,05	9,32

Tabela G.6 - Tempo de *delete* para amostra de 10 testes com variação de CPU

Tempo de Delete para amostra de 10 testes														
SO	Memória	Disco	CPU	Concorrência	#01	#02	#03	#04	#05	#06	#07	#08	#09	#10
Ubuntu	4096	60	2	2	2,47	4,39	2,42	2,36	2,30	4,44	4,48	2,44	4,44	4,42
Win7	4096	60	2	2	2,35	4,50	2,46	2,55	4,62	2,56	4,37	4,68	4,59	2,77
Win8	4096	60	2	2	2,32	4,54	2,39	4,61	2,47	2,38	4,46	4,55	4,67	2,34
Ws2012	4096	60	2	2	4,37	4,48	2,34	4,38	4,48	4,48	2,84	4,48	4,55	2,42
Ubuntu	4096	60	4	2	2,39	4,38	2,45	4,69	2,37	4,70	2,40	4,57	4,44	4,48
Win7	4096	60	4	2	2,34	4,66	2,34	4,42	4,43	2,56	2,34	4,49	4,86	4,83
Win8	4096	60	4	2	4,35	4,40	2,37	4,38	4,50	2,41	2,59	4,58	2,38	4,58
Ws2012	4096	60	4	2	2,42	4,41	4,42	4,47	2,46	4,38	4,55	2,45	2,83	4,87
Ubuntu	4096	60	8	2	4,40	4,39	2,48	2,37	4,76	4,54	2,87	4,49	4,68	4,82
Win7	4096	60	8	2	2,91	4,47	4,89	4,85	5,05	5,02	2,37	4,82	4,49	4,37
Win8	4096	60	8	2	4,40	4,40	2,32	4,44	4,49	4,40	4,54	4,41	4,40	4,49
Ws2012	4096	60	8	2	4,40	2,61	2,36	4,52	4,62	4,53	4,89	4,44	4,48	4,55
Ubuntu	4096	60	2	4	4,55	4,82	4,60	5,02	2,73	4,57	4,82	4,69	2,31	4,64
Win7	4096	60	2	4	4,50	4,54	4,56	4,56	5,22	5,37	5,40	4,57	4,45	2,46
Win8	4096	60	2	4	4,54	4,73	4,79	5,02	4,50	4,68	4,42	4,67	4,38	4,47
Ws2012	4096	60	2	4	4,53	4,79	4,81	4,54	4,46	4,76	4,64	4,62	4,54	4,44
Ubuntu	4096	60	4	4	4,60	4,54	4,38	4,66	3,29	5,10	5,11	4,65	2,33	4,39
Win7	4096	60	4	4	4,50	4,52	4,82	4,94	4,40	4,79	4,59	4,73	2,49	4,40
Win8	4096	60	4	4	4,53	4,62	4,59	4,96	6,71	4,59	7,03	6,89	2,84	4,87
Ws2012	4096	60	4	4	4,81	4,75	4,75	4,60	4,47	4,51	4,75	4,98	4,49	4,48
Ubuntu	4096	60	8	4	4,81	4,85	4,73	5,28	4,72	4,88	4,94	4,47	4,44	4,54
Win7	4096	60	8	4	4,55	4,68	4,68	4,58	6,50	4,82	4,39	4,55	2,47	4,68
Win8	4096	60	8	4	4,85	4,62	4,54	4,63	4,61	4,66	4,72	4,84	4,53	4,88
Ws2012	4096	60	8	4	2,79	4,95	4,62	4,67	6,67	4,82	4,86	4,58	4,41	4,38
Ubuntu	4096	60	2	6	6,72	5,19	4,89	4,70	7,68	7,70	2,93	4,94	4,79	4,59
Win7	4096	60	2	6	6,92	5,16	5,28	5,29	4,87	6,91	4,65	4,50	4,53	4,54
Win8	4096	60	2	6	6,68	4,78	5,58	7,12	7,73	7,73	4,46	4,84	4,81	4,59
Ws2012	4096	60	2	6	4,76	5,24	4,57	7,32	7,36	6,60	4,50	4,49	4,49	4,50
Ubuntu	4096	60	4	6	4,51	4,41	5,83	5,86	7,28	7,33	4,64	5,18	4,50	4,43
Win7	4096	60	4	6	6,69	6,94	5,21	7,15	4,77	6,94	6,54	4,75	4,55	4,53
Win8	4096	60	4	6	7,06	4,88	4,55	6,65	7,24	7,29	4,50	4,57	4,67	2,43
Ws2012	4096	60	4	6	6,67	4,91	5,36	5,32	7,37	6,88	4,56	4,45	4,97	4,85
Ubuntu	4096	60	8	6	7,06	6,72	5,26	5,09	6,51	7,44	2,36	4,51	4,99	5,00
Win7	4096	60	8	6	5,10	5,35	5,39	4,54	8,06	7,88	4,53	4,81	4,61	4,44
Win8	4096	60	8	6	6,87	4,80	6,62	6,98	7,31	6,94	6,91	4,49	4,47	4,55
Ws2012	4096	60	8	6	6,72	4,93	4,63	7,02	6,85	7,12	4,68	4,40	4,67	4,64