Replicação e Caching num Sistema Moderno de Virtualização de Desktops

Luís Silva, Paulo Lopes, Nuno Preguiça, Pedro Medeiros, João Leitão[1], and Miguel Martins[2]

[1] NOVA LINCS & DI, FCT, Universidade NOVA de Lisboa
[2] SolidNetworks – Business Consulting, Lda

Resumo O grande aumento da utilização de técnicas de virtualização levou à alteração da forma como são geridas as máquinas físicas, bem como os modelos das infra-estruturas de um centro de dados. Encontram-se em constante desenvolvimento tecnologias relevantes que abordam problemas como a virtualização de desktops - todas com o seu mérito, mas ignorando completamente o poder de computação presente nos PC das organizações, optando por uma via de elevado custo baseada em servidores poderosos. Neste artigo apresenta-se uma arquitectura de uma solução de virtualização de desktops com a computação baseada nos clientes, que inclui um mecanismo de replicação e caching para a distribuição do armazenamento das imagens das máquinas virtuais. Este mecanismo explora um sistema de ficheiros local, com suporte a snapshots e gere eficientemente ficheiros com vários GB acedidos maioritariamente em modo de leitura mantendo destes múltiplas versões. A solução desenvolvida oferece desempenho, alta disponibilidade e escalabilidade na presença de um elevado número de clientes geograficamente distribuídos, apresentamos uma avaliação inicial desta solução, efectuada num cenário de produção, que mostra ir ao encontro destas características.

Palavras-chave: Middleware de Replicação e Caching · Infraestrutura de Desktop Virtual · Sistema de Ficheiros com Snapshots.

1 Introdução

O conceito de virtualização não é recente [13], apesar de todas as recentes discussões e novidades. Com o avanço de técnicas que aproveitam eficientemente todos os recursos disponíveis e uma notória redução nos custos de aquisição de hardware, surge a oportunidade para o desenvolvimento de aplicações com novas arquitecturas e uma reformulação nas infra-estruturas que lhes fornecem suporte. [6]

Algumas aplicações possuem peculiaridades - como a necessidade de baixa latência e um grande espaço de armazenamento, determinando que o poder de computação e dados devam estar o mais próximo possível do seu consumidor. Um destes casos é a Virtualização de Desktops. As infra-estruturas que suportam este

tipo de aplicação, vulgo, *Virtual Desktop Infrastructures* (VDI) são estruturas de computação que disponibilizam aos seus utilizadores desktops virtuais: quando um utilizador inicia o seu posto de trabalho e interage com ele, as aplicações que são mostradas no ecrã do utilizador não são executadas nativamente na workstation, correm numa máquina virtual (VM).

Neste artigo mostramos uma plataforma VDI, que se destaca das soluções actuais (apelidadas *Server-based* VDI), que executam os desktops virtuais em poderosos servidores com amplos recursos. Sendo a nossa abordagem (apelidada *Client-based* VDI), suportada directamente nas workstations dos utilizadores de uma maneira não intrusiva (sem utilizar armazenamento persistente local), oferecendo desktops nativos (praticamente qualquer distribuição Linux) e virtuais (qualquer Sistema Operativo (SO) "virtualizável"). E tal como acontece na Server-based VDI qualquer dos diferentes SOs disponibilizados é utilizável de imediato em qualquer workstation (dispensa prévia instalação local), assim diferentes PCs podem estar a executar diferentes SOs.

A importância de uma estrutura de armazenamento de dados distribuída leva à necessidade de um mecanismo de replicação que não só funcione como uma rede de segurança em casos de falhas (uma vez que a duplicação de dados elimina a característica de existir um ponto único de falha), mas também é utilizada de modo a aproveitar a largura de banda da rede (um acesso pode consultar vários servidores). Viabilizando assim propriedades como a tolerância a falhas, alta disponibilidade e escalabilidade na solução.

Este artigo encontra-se organizado da seguinte forma. A Secção 2 motiva este trabalho analisando soluções existentes e discutindo as suas vantagens e limitações. De forma a possibilitar uma melhor compreensão de toda a plataforma a Secção 3 apresenta uma visão sobre as principais características do projecto. Segue-se, na Secção 4 uma descrição arquitectural da plataforma e a introdução de um dos problemas resolvidos. A apresentação e descrição da solução é feita em detalhe na Secção 5, com os resultados da nossa avaliação experimental apresentados na Secção 6. Por fim a Secção 7 conclui o artigo salientando as ideias mais importantes.

2 Trabalho Relacionado

Infra-estrutura de Desktops Virtuais (VDI) O conceito de virtualização de desktops engloba uma série de técnicas, que juntas, facilitam a disponibilização *on-demand* de desktops, onde todo o software necessário se encontra agregado numa única unidade - uma máquina virtual; é nessa que toda a computação ocorre. Este mecanismo permite a virtualização de um posto de trabalho que pode ser executada num típico PC / Laptop, ou num servidor. Estas soluções por norma apresentam uma estrutura centralizada, com o ambiente de cada utilizador a residir num sistema instalado num centro de dados, bem como outros componentes necessários, como armazenamento para os utilizadores e dados das máquinas virtuais. É ainda necessária uma rede capaz de transportar grandes blocos de dados

de forma expedita, tendo em conta que na perspectiva do utilizador não deve existir diferenças aparentes entre uma instalação local e um desktop virtual.

Existem várias soluções comerciais e em desenvolvimento que implementam plataformas de virtualização de desktops, podendo estas ser caracterizadas em vários tipos - desde soluções inteiramente na Nuvem (correntemente denominada de Desktop-as-a-service, ou abreviadamente DaaS), a soluções que tiram proveito de uma infraestrutura local ou híbrida.

Desktops Virtuais baseados em Servidores Ou *Server-based VDI*, é a abordagem mais comum, sendo aquela em que uma VM é executada remotamente num servidor com o recurso a um hipervisor. Neste modelo, as imagens utilizadas são armazenadas num Centro de Dados, e quando chega a altura de serem postas em execução, um servidor por meio de um hipervisor disponibiliza o ambiente para a sua execução. Esta abordagem baseada em servidores, ostenta benefícios pelo facto de permitir a utilização de *thin* e *zero clients* de muito baixa performance que apenas necessitam de suportar protocolos como o *Remote Desktop Protocol (RDP)* [3] ou o *Remote Framebuffer Protocol (RFB)* [16] para a interacção com o desktop virtual. Pelo lado negativo existe o custo de uma plataforma deste tipo, sendo necessária uma infra-estrutura com grande capacidade (computação, armazenamento, rede e energia). Mais ainda, pode-se observar que a capacidade de computação local, na prática muitas vezes apreciável, que habitualmente já se encontre instalada não é aproveitada por este tipo de plataforma. Estas máquinas naturalmente têm recursos para usar as ferramentas mencionadas (RDP e RFB), mas o não aproveitamento de todo o seu potencial faz com que grande parte do investimento em hardware feito no passado seja desperdiçado. As principais soluções comerciais disponíveis são a plataforma Horizon [4] da VMware, XenDesktop [5] da Citrix e a Microsoft com Microsoft Remote Desktop [1].

Desktops Virtuais baseados em Clientes Ou Client-based VDI, é um modelo em que uma máquina virtual, que alberga o desktop virtual é executada directamente no posto de trabalho do cliente. Esta máquina faz uso de um hipervisor que lida por inteiro com o desktop virtual. Uma vez que o trabalho de computação predomina no lado do cliente, a capacidade de processamento e memória central da infra-estrutura de suporte necessária (no que toca aos servidores) neste modelo é muito inferior, tendo apenas como trabalho principal a gestão do ambiente de armazenamento. Em alternativa, os dados até poderiam estar armazenados nos discos dos clientes, o que relegaria os servidores para um papel puramente administrativo e de manutenção de outros serviços. As vantagens são semelhantes às apresentadas na solução anterior Server-based VDI, com o benefício adicional de os recursos necessários serem menores, possibilitando uma maior rentabilização de equipamento já instalado na infra-estrutura. Embora esta solução seja vantajosa em termos de custos, não se verifica um grande interesse no seu desenvolvimento. O XenClient [5] da Citrix é a única solução que chegou ao mercado, mas foi abandonada no final de 2015, apenas cinco anos após ter sido lançado. Razões para este facto podem ser atribuídas a um processo complicado de desenvolvimento, e à reserva para uso exclusivo dos

discos rígidos das estações de trabalho, com consequente apagamento dos dados pré-existentes. [2]

Desktops como um Serviço (DaaS) Este ultimo conceito, sendo o mais moderno, incorpora uma arquitectura VDI com os mais recentes serviços cloud, sendo em muitos aspectos muito semelhante à solução *Server-based*, onde os servidores são responsáveis pela computação, mas aqui, a infra-estrutura, os recursos e a gestão encontra-se toda numa cloud publica. Existem pontos positivos claro - um óptimo potencial na redução de custos da aquisição de equipamentos, uma vez que a infra-estrutura é gerida por terceiros, incluido questões como a sua manutenção, segurança dos dados e flexibilidade no aprovisionamento de recursos. Em contraste, as desvantagens também são algumas. Como os dados encontram-se armazenados num local distante do seu consumo, surgem problemas relacionados com a largura de banda que pode limitar o número de conexões. Adicionalmente observa-se outro problema, este incontornável, a latência resultante da distancia e da velocidade a que os dados podem percorrer essa distancia; e o *jitter*, causado pelas variações na latência, que podem ser observadas quando as conexões têm que percorrer largas distancias atravessando vários *providers* e diferentes taxas de congestionamento. Neste campo, existe uma multiplicidade de soluções oriundas dos fornecedores de serviços cloud. A Amazon, no seu portefólio AWS, oferece o Amazon Workspaces; a Microsoft implementa o RDS na linha de produtos Azure e mais recentemente apresentaram o serviço Windows Virtual Desktop que ainda se encontra em fase de pré-lançamento; a VMware conta com o Horizon Cloud para disponibilizar a sua infra-estrutura, tendo firmado parcerias com a IBM e a Microsoft; existem mais soluções nesta corrida mas com menor expressão, como é o caso da Citrix com o Citrix Workspace e da Workspot (uma empresa fundada por ex-funcionários da Citrix).

3 Visão Geral

O acrónimo iCBD significa *Infrastructure for Client-Based (Virtual) Desktop (Computing)* [11], tem como principal objectivo implementar uma plataforma de virtualização de desktops centrada no cliente, apresentando-se portanto, como uma forma especializada de VDI. Neste caso, todas as tarefas relativas à computação - desde a exibição de um ambiente gráfico à execução de aplicações - são realizadas directamente no hardware da workstation de um utilizador (seja um PC, Laptop, etc.), ao contrário de ser efectuada em servidores poderosos e caros, tal como se referiu antes com as implementações correntes de VDI, disponibilizadas pela Citrix, Microsoft ou VMware, entre outras.

Resumindo, o objectivo é preservar a conveniência e simplicidade de uma plataforma de gestão completamente centralizada para desktops Linux e Windows, instanciando esse desktops nas máquinas físicas dos clientes por meio de *templates* de máquinas virtuais armazenadas em repositórios.

Este projecto apresenta alguns aspectos de vanguarda das actuais plataformas de virtualização de desktops que se fazem sobressair, como a adopção de

um paradigma *diskless* com boot remoto, a total preservação dos dados pré-existentes nos eventuais discos locais, a forma como as imagens de maquinas virtuais são armazenadas na plataforma, e o suporte para uma execução virtualizada ou nativa (apenas para clientes Linux), consoante a escolha do utilizador. O processo de boot remoto do ambiente de trabalho de um utilizador requer a cooperação de serviços como HTTP, TFTP, DHCP e os servidores repositórios de imagens - tanto os que disponibilizam os templates de maquinas virtuais em modo de leitura como os responsáveis por disponibilizar o espaço de escrita para as instancias em execução que foram criadas a partir dos mencionados templates. No processo de boot as comunicações entre a plataforma e as estações de trabalho efectuam-se através do protocolo HTTP, providenciando tanto flexibilidade como eficiência.

De forma a simplificar a terminologia, formulámos o termo *iCBD Machine Image* (iMI), que engloba os ficheiros necessários à inicialização de um cliente iCBD. Uma iMI inclui um template de uma máquina virtual (contanto com um sistema operativo, configurações e aplicações), o pacote de boot iCBD (uma colecção de ficheiros necessários ao boot com recurso à rede e adaptado ao sistema operativo) e um compêndio de configurações para serviços como PXE e iSCSI.

4 Arquitectura

A plataforma iCBD é composta por uma multitude de serviços, que podem ser agrupados em quatro grandes blocos arquitecturais.

Camada de Serviços de Boot é responsável por fornecer o código inicial que suporta o boot por rede das máquinas cliente, fazendo a transferencia de um pacote feito por medida (SO, ramdrive, scripts iniciais), utilizando serviços como PXE, DHCP, TFTP e HTTP.

Camada de Suporte ao Cliente fornece o suporte para as operações do lado do cliente, incluindo, autenticação, espaço de armazenamento de leitura e escrita para as instancias de clientes (uma vez que as iMIs são executadas em workstations sem utilizarem o disco fisico) e acesso às directorias *home* dos utilizadores.

Camada de Administração mantém todo o ciclo de vida das iMIs, desde a sua criação até à sua reforma. Neste momento, o processo é efectuado por um conjunto especifico de scripts.

Camada de Armazenamento mantém o repositório de iMIs e fornece algumas operações essenciais como o controlo de versões dos ficheiros de imagens de maquinas virtuais. Na secção 3 desenvolvemos mais acerca de trabalho realizado nesta camada, estendendo de modo a que um repositório seja abstraído e possa ser implementado sobre vários repositórios locais através de um processo de replicação e caching. Estes repositórios locais são implementados sobre Btrfs [15] ou Ceph [17] e podem ser exportados para os clientes utilizando NFS, iSCSI, REST ou outros protocolos que sejam adequados.

Parte mais significante do trabalho aqui apresentado tem o seu foco nesta camada, que é responsável por armazenar todos os dados da plataforma, sejam estes iMIs, discos virtuais (.vmdk) para maquinas virtuais como as de administração, bem como outros, ligados aos serviços que suportam a plataforma - bases de dados que guardam informação e configurações desses serviços, repositórios de código, etc.. No fundo, a camada de armazenamento é composta por um conjunto de sistemas de ficheiros (ou de *object stores*, no caso de serem *object-based*) cada um com o seu propósito. Neste artigo damos enfoque no sistema de ficheiros que faz o armazenamento das iMIs, uma vez que este tipo de dados é muito particular, e o sistema de ficheiros tem que dispor de um dado conjunto de funcionalidades, sendo de particular importância o suporte de snapshots, fazendo cair a nossa escolha no Btrfs. [7] [12]

As funcionalidades do Btrfs mais utilizadas na plataforma iCDB são: sub-volumes; snapshots; clonagem de sub-volumes e snapshots; *Btrfs seeding*; backups incrementais, apenas para mencionar alguns. Sendo que, cada funcionalidade tem a sua finalidade: vários sub-volumes são utilizados para armazenar os dados das diferentes partes da plataforma; os snapshots são amplamente utilizados para criar (um tipo de) controlo de versões das iMIs (possibilitando o acesso a qualquer versão a qualquer momento) e como uma forma de fazer backup de toda a plataforma. Com recurso ao *Btrfs seeding* possibilita-se montar várias vezes o mesmo sistema de ficheiros, em modo de leitura, proporcionando que clientes usem a mesma iMI.

Pelo ponto de vista das restantes camadas, a forma como os dados são armazenados deve ser totalmente transparente, uma vez que apenas vão interagir com protocolos standard como o NFS e iSCSI. Por conseguinte, ao lidar com o problema da replicação de dados por entre vários sistemas de ficheiros, a transparência, como atributo, é fundamental. Discutiremos em detalhes os tópicos de replicação e armazenamento de dados em cache na próxima secção.

5 Sistema de Replicação e Caching

Um dos principais objectivos do projecto é fornecer uma plataforma com uma arquitectura que se estenda desde totalmente alojada *on-site*, até uma onde grande parte dos serviços estejam instalada numa cloud publica. Naturalmente, soluções intermédias também são interessantes, em que, por exemplo, a administração e armazenamento das *"golden"* iMIs são baseados na nuvem, enquanto o restante se encontra *on-premises*, junto dos clientes. Portanto, torna-se evidente que a localização dos dados é um tema de particular interesse, sendo necessário estudar os fluxos de dados entre os vários componentes do iCBD - uma plataforma complexa, com vários dispositivos de armazenamento e diferentes redes, que são acedidas por vários clientes que necessitam desses dados a qualquer instante.

Todos estes factores, aliados a um desejo de ter uma plataforma com uma arquitetura distribuída e escalável, resultam na necessidade de projetar um serviço cuja missão é garantir que os dados sejam replicados corretamente, nos locais

apropriados, e que mantenham a consistência das várias versões das iMIs. O que resultou na arquitetura de alto nível apresentada na Figura 1.

O serviço de replicação e caching (RCS) necessita de interagir com diferentes ferramentas, por exemplo, *scripts* em *bash*, ferramentas da linha de comandos e algumas chamadas ao *kernel* do SO, determinou-se então que a abordagem mais adequada seria criar uma camada de *middleware* distribuída em Python utilizando um paradigma *master-replica*.

O nosso *middleware* é composto por vários módulos e inclui todas as funcionalidades necessárias a permitir que um nó se comporte como *master* ou *replica*, sendo que, cada um mantém um repositório de iMIs individual. Neste processo foram também desenvolvidas várias bibliotecas: para interagir com ferramentas como o Btrfs *send / receive* e com o SSH, e *wrappers* para alguns algoritmos de compressão e até mesmo para fornecer uma API REST.

Segue-se uma visão geral com uma pequena descrição das funcionalidades dos principais componente do sistema RCS, é necessário referir que estes não são os únicos componentes deste serviço, apenas os mais importantes para compreender o seu funcionamento.

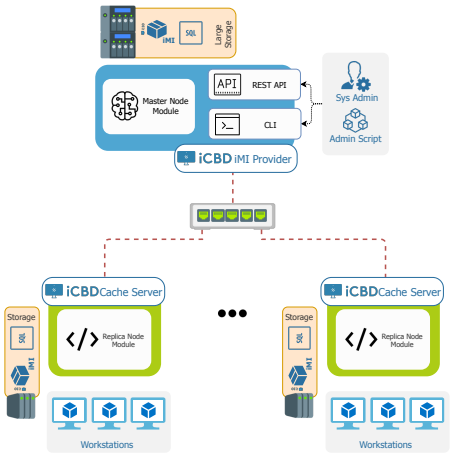

Fig. 1: Arquitectura do Serviço de Replicação e Caching (alto nível)

O *Master Node* actua como um controlador para o sistema de replicação e caching e como uma interface para a interação com um cliente, seja por meio de uma CLI ou da API REST. O *Replica Node* tem como tarefa principal manter a lista de iMIs que não estejam atualizadas, receber as atualizações assim que o Master Node as enviar e armazená-las localmente. O *Name Server* é um serviço que reside na mesma máquina que o Master Node, contém registros sobre os nós presentes no sistema de replicação e caching, numa base de dados simples e similar a uma lista telefónica. O *Image Repository* é uma estrutura de dados concebida à medida para conter um grande conjunto de objectos que rep-

resentam a localização das iMIs numa `key:value` *store*. E a biblioteca *Btrfs Lib* possui duas classes: a primeira, chamada `Btrfs Tool`, é um *wrapper* em Python para a ferramenta `btrfs-progs`; a outra classe, designada `BtrfsFsCheck`, implementa funções que validam se um determinado caminho pertence a um sistema de ficheiros Btrfs, e se tal se confirmar, verifica se nesse caminho existe um sub-volume. Mais ainda, é disponibilizado um método para descobrir todos os snapshots presentes num diretório.

5.1 Processo de Replicação

Uma vez que o sistema de ficheiros base que é utilizado pela plataforma tem o suporte para snapshots, queremos retirar o máximo partido das vantagens desta funcionalidade e minimizar a quantidade de dados que necessitam de atravessar a rede. O Btrfs disponibiliza um conjunto de ferramentas que podem ser utilizadas em *userspace* e que possibilitam a gestão do sistema de ficheiros, este utilitário chamado `btrfs-progs`, inclui um par de comandos `btrfs send` [10], e `btrfs receive` [9], que fornecem a capacidade de transportar dados por meio de *stream* e aplicar as diferenças de/para um sistema de ficheiros remoto.

Envio de um Snapshot para um Cache Server (Replica Node) O processo segue estas etapas: quando uma tarefa de administração de uma iMI é concluída, o Master Node é notificado da existência de uma nova versão dessa iMI; uma nova entrada para essa versão é criada e armazenada no Image Repository local; seguidamente, o procedimento principal de replicação é iniciado.

Findos estes preliminares, o procedimento de replicação opera da seguinte forma: primeiro, é compilada a lista de Replica Nodes que sobrescreveram o seu interesse nessa iMI; de seguida, e para cada nó, obtém-se a ultima versão da iMI disponível no seu Image Repository, e esse "valor" é usado para determinar se a nova versão pode ser enviada imediatamente ou se é necessário transferir algumas versões intermédias. Em qualquer dos casos, apenas as diferenças entre as versões são transmitidas, não a iMI completa. Supondo que apenas o ultimo delta (as alterações mais recentes) será enviado, é necessário decidir se a transferência ocorrerá usando um canal de comunicação simples ou cifrado, e/ou se algum algoritmo de compressão será aplicado aos dados antes do seu envio. Toda esta informação é transmitida em antecedência ao nó receptor para que esteja pronto a acolher os dados.

O comando `btrfs send`, mencionado acima, simplifica o processo de geração de um stream de instruções que descrevem as alterações entre dois snapshots de um sub-volume. Adicionalmente, este comando apresenta a capacidade de usar um modo incremental, onde um dado snapshot pai que esteja presente em ambos os lados da transferência é utilizado como base e apenas as alterações entre esse e o novo snapshot integram o stream de dados. Este método reduz consideravelmente a quantidade de dados a ser transferida e necessária à reconstrução do snapshot no lado receptor. As operações desta ferramenta no lado do envio ocorrem no kernel, começando por determinar as diferenças entre os sub-volumes, e,

baseando-se nessas diferenças o kernel gera um conjunto de instruções que constituem um stream personalizado que vai ser descodificado no lado da recepção.

Recepção de um Snapshot O processo para receber uma versão de um iMI é complementar ao processo de envio explicado acima, embora muito mais simples. Conforme explicado, a operação `send` necessita de alguns recursos computacionais, pois precisa calcular todas as diferenças entre as versões para transferir e, após essa etapa, criar um stream de operações que, quando executadas, recriam precisamente as diferenças entre as versões num novo snapshot. Portanto, no lado da recepção desse stream (neste caso, o Replica Node), o comando `receive` aceita o stream gerado e usa esses dados para recriar um ou mais snapshots. Ao contrário da operação `send`, o `receive` é executado em *userspace*, reproduzindo as instruções provenientes do stream uma por uma; o conjunto de instruções inclui as chamadas mais relevantes encontradas num Virtual File System, como `create()`, `mkdir()`, `link()`, `symlink()`, `rename()`, `unlink()`, and `write()`, entre outras [8].

6 Avaliação

6.1 Configuração Experimental

A infraestrutura iCBD no DI - FCT NOVA é composta por um cluster de dois nós baseado em servidores HPE ProLiant DL380 Gen9 conectados por meio de um switch HPE Flexfabric 5700 com links a 10Gbps, e com armazenamento externo fornecido por um HPE MSA 2040 SAN Storage Disk Array.

No que toca a clientes, utilizados na validação funcional e testes de performance, foram gentilmente cedidos dois laboratórios (Laboratórios 110 e 112) do DI - FCT NOVA cada um com quinze PCs modernos, tendo ligações gigabit e suporte a Intel VT [14], portanto capazes de executar hipervisores.

Para os nós "remotos", ou seja, os servidores de replicação e cache, nos nossos testes, foram utilizados dois tipos de servidores: virtuais lançados no cluster e um físico. Este ultimo, um desktop recondicionado de características modestas, com a particularidade de se encontrar ligado ao mesmo switch que os clientes do Lab. 112. Enquanto que as comunicações com os clientes do Lab.110 têm que atravessar alguns switches na rede da Faculdade e, portanto, podem sofrer de condições adversas na rede, estando totalmente fora do nosso controlo.

6.2 Validação Funcional

Para testar a correção do funcionamento do módulo de replicação num ambiente com vários nós, foi iniciado um Master Node e três Replica Nodes. Observámos que os Replica Nodes se registraram no Name Server, conforme o esperado, e que as comunicações entre o Master Node e as Réplicas também se realizaram correctamente. Em seguida, executamos repetidamente dois tipos de testes: um focado no envio de uma versão completa de uma iMI (uma que não estava presente no Image Repository das Réplicas), forçando os dados a serem transferidos

pela rede. Além disso, queríamos verificar que, quando a transferência fosse concluída, o Image Repository local fazia reflectir a adição da nova iMI. Este cenário é comum quando uma Réplica subscreve uma nova iMI.

O teste funcional do servidores de cache foi realizado com recurso ao servidor de cache físico, particularmente porque este se encontra conectado diretamente a um dos laboratórios e, assim, permite o teste imediato do boot de uma iMI num posto de trabalho. Dado o grau de integração da plataforma iCBD com os demais serviços e políticas de rede da FCT NOVA, foi necessário um considerável processo de experimentação, ajustando alguns parâmetros até chegarmos a uma configuração totalmente funcional. No final, foi confirmado que era possível iniciar as workstations com iMIs provenientes do cache server.

6.3 Avaliação do processo de replicação

Para esta avaliação utilizou-se uma iMI Linux como a distribuição Ubuntu Desktop 16.04 LTS, ocupando 38,60 GB, e foram criados quatro cenários:

O primeiro, *Rsync*, representa uma transferência de uma iMI usando a ferramenta rsync [3] com as opções `-r` (recursão em diretórios); `-t` (preservar datas de modificação); `-p` (preservar permissões); `-l` (copiar links simbólicos como links simbólicos) e `-u` (ignorar arquivos que são mais recentes no lado de recepção). O *iCBD-Rep - I* utiliza o Serviço de replicação do iCBD para transferir uma iMI usando os sockets padrão do Python e sem compressão. O *iCBD-Rep - II* usa o Serviço de replicação do iCBD para transferir uma iMI usando os sockets padrão do Python e compressão com o algoritmo LZ4. E o *iCBD-Rep - III* serve-se do Serviço de replicação do iCBD para transferir uma iMI usando uma conexão SSH e sem compressão.

Envio de uma iMI completa: Este teste envia uma iMI completa, ou seja, neste caso todos os seus 38,60 GB de dados; observamos que nosso módulo de replicação comporta-se de forma semelhante nos três testes (I a III), mas fica aquém quando comparado ao rsync; os resultados são mostrados na Tabela 1.

Mostra-se também que o processo de transferência pelo rsync é mais rápido que o nosso utilizando o método Python-mais-btrfs-send; no entanto, o uso do rsync tem uma desvantagem: concluída a transferência é necessário usar as ferramentas do Btrfs para recriar a estrutura da iMI e isso leva tempo (o que não medimos).

Envio de um delta entre duas versões de uma iMI: Neste teste criou-se uma nova versão da iMI referida no teste anterior, tendo sido administrada e instaladas as actualizações sugeridas pela ferramenta `apt`, esta operação resultou num delta com o tamanho de 4.45 GB. Apenas esses dados foram transferidos neste teste.

[3] É importante notar que a ferramenta rsync não consegue discernir o que é um subvolume Btrfs, portanto os tempos apresentados referem-se apenas à transmissão de dados, mas seriam necessárias mais operações para tornar a iMI funcional dentro da plataforma iCBD.

	Tempo	Dados (MB)
Rsync	12m23s	39543
iCBD-Rep - I	17m21s	38947
iCBD-Rep - II	20m35s	35572
iCBD-Rep - III	22m55s	39412

Tabela 1: Tempo gasto e dados transmitidos na transferência de uma iMI completa do Master para a Réplica

Logo à partida, pode-se ver, como esperado, uma redução drástica na quantidade de dados transmitidos pela rede. Os resultados são mostrados na Tabela 2.

	Tempo	Dados (MB)
Rsync	10m15s	5964
iCBD-Rep - I	1m47s	4864
iCBD-Rep - II	2m15s	4713
iCBD-Rep - III	2m55s	4902

Tabela 2: Tempo gasto e dados transmitidos na transferência de um delta de uma iMI do Master para a Réplica

Nossa solução usando as operações Btrfs `send` e `receive` mostram-se muito superiores ao desempenho do rsync no mesmo conjunto de dados. Mesmo nos casos em que as opções de compressão ou de cifra foram usadas, os resultados não mudaram drasticamente. Aqui, a nossa conclusão para a diferença impressionante entre o rsync e nossa solução é que o rsync leva uma quantidade enorme de tempo a computar as diferenças (com comparações bloco a bloco) e a decidir sobre o que deve enviar.

7 Conclusões

Neste artigo apresentamos uma plataforma VDI centrada nos clientes, que se denota das demais pela sua abordagem, em que fornece suporte directamente nas workstations dos utilizadores de uma maneira não intrusiva, oferecendo desktops nativos e virtuais utilizáveis de imediato.

Mostrámos que nossa implementação de um sistema de replicação para iMIs com base num modelo de Master - Replica fornece um processo eficiente para distribuir as iMIs para os seus destinos num ambiente multi-servidor, e adicionalmente garante algum grau de tolerância a falhas à plataforma iCBD. Contando com a camada de controlo do RCS, que simplifica todo o processo mantendo o estado dos nós.

Finalmente, avaliámos os componentes dedicados ao processo de replicação e cache por meio de uma validação funcional abrangente e criamos um bench-

mark para avaliar o desempenho deste sistema. Na nossa opinião, os resultados demonstram claramente os benefícios das soluções propostas.

Agradecimentos Este trabalho foi parcialmente suportado pelo financiamento plurianual do programa *COMPETE2020 / PORTUGAL2020* do projecto *iCBD* (POCI-01-0247-FEDER- 011467).

Referências

1. Microsoft remote desktop services (rds) explained (2010), https://technet.microsoft.com/en-us/video/remote-desktop-services-rds-explained.aspx
2. Citrix bids adieu to xenclient (2015), http://vmblog.com/archive/2015/09/24/citrix-bids-adieu-to-xenclient.aspx
3. Remote desktop protocol (2017), https://msdn.microsoft.com/en-us/library/aa383015(v=vs.85).aspx
4. Vmware horizon (2017), http://www.vmware.com/products/horizon.html
5. Xenapp & xendesktop (2017), https://www.citrix.co.uk/products/xenapp-xendesktop/
6. Agesen, O., Garthwaite, A., Sheldon, J., Subrahmanyam, P.: The evolution of an x86 virtual machine monitor. SIGOPS Oper. Syst. Rev. **44**(4), 3–18 (Dec 2010)
7. Alves, N.: Linked clones baseados em funcionalidades de snapshot do sistema de ficheiros. Master's thesis, Universidade NOVA de Lisboa (2016)
8. Kernel.org - Linux Kernel Organization, Inc: Design notes on Send/Receive - btrfs Wiki (2018), https://btrfs.wiki.kernel.org/index.php/Design_notes_on_Send/Receive
9. Kernel.org - Linux Kernel Organization, Inc: Manpage/btrfs-receive - btrfs Wiki (2018), https://btrfs.wiki.kernel.org/index.php/Manpage/btrfs-receive
10. Kernel.org - Linux Kernel Organization, Inc: Manpage/btrfs-send - btrfs Wiki (2018), https://btrfs.wiki.kernel.org/index.php/Manpage/btrfs-send
11. Lopes, P., Preguiça, N., Medeiros, P., Martins, M.: icbd: Uma infraestrutura baseada nos clientes para execução de desktops virtuais. In: Proceedings CLME2017/VCEM 8º Congresso Luso-Moçambicano de Engenharia/V Congresso de Engenharia de Moçambique. pp. 13–18 (2017)
12. Martins, E.: Object-Base Storage for the support of Linked-Clone Virtual Machines. Master's thesis, Universidade NOVA de Lisboa (2016)
13. Popek, G.J., Goldberg, R.P.: Formal Requirements for Virtualizable Third Generation Architectures. Communications of the ACM **17**(7), 412–421 (1974)
14. Righini, M.: Enabling Intel Virtualization Technology Features and Benefits. Tech. rep. (2010), https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/virtualization-enabling-intel-virtualization-technology-features-and-benefits-paper.pdf
15. Rodeh, O., Bacik, J., Mason, C.: BTRFS: The Linux B-Tree Filesystem. ACM Transactions on Storage **9**(3), 1–32 (2013)
16. T. Richardson, J.L.: The remote framebuffer protocol. RFC 6143, Internet Engineering Task Force (IETF) (March 2011), https://tools.ietf.org/html/rfc6143
17. Weil, S.A., Brandt, S.A., Miller, E.L., Long, D.D.E., Maltzahn, C.: Ceph: A Scalable, High-Performance Distributed File System. Proceedings of USENIX Symposium on Operating Systems Design and Implementation pp. 307–320 (2006)