FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Módulo para Gestão de Atualizações de *Software IPBrick*

Francisco Oliveira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Ricardo Morla

Co-orientador: Miguel Ramalhão

26 de Julho de 2016

Resumo

Esta dissertação consiste na idealização e implementação de um Módulo para Gestão de Atualizações do *Software IPBrick*. A *IPBrick* trata-se de uma empresa com soluções empresariais de Comunicações Unificadas sobre o Protocolo da Internet estas soluções consistem num sistema operativo e aplicações como o *CAFE* e o *iPortalDoc*, que são uma rede social de formato empresarial e um gestor de documentos, respetivamente. A Empresa iniciou-se à relativamente pouco tempo no mundo tecnológico da *Cloud*. Nesse sentido, surgiu a necessidade de criar um módulo que permitisse ao seu utilizador atualizar o sistema e as suas aplicações com o simples clique do botão de um rato ou então com definições que garantem a automatização do processo. O módulo deve adaptar-se às definições impostas pelo utilizador e estar concordante com as limitações existentes no *software* da empresa e no sistema operativo que lhe serve de base, o *Linux Debian*.

O objetivo e motivação do módulo podem ser explicados através de uma analogia simples. Pense-se num carro de uma marca de uma certa série e cilindrada, este carro possui inúmeras partes que o proprietário pretende manter mas este também gostava que pudesse passar o seu carro para a série seguinte ou que pudesse manter a série mas aumentar a cilindrada. A Empresa pretende dar resposta a isto através de uma oficina onde se mantém tudo o que o cliente pretende mas substituindo algumas peças consegue-se oferecer o tipo de carro que este pretende também.

Em suma, a implementação descrita baseia-se num módulo dividido em duas partes: *major* e *minor updates*. Onde o segundo ainda se encontra a ser implementado e onde o primeiro teve um processo criado de raiz com o nome *SRRDR* que permite fazer uma atualização de distribuição sem paragem total de serviços, em *background* através da aglutinação do espaço livre existente em disco após reestruturação das partições. Foi possível reduzir bastante o tempo de instalação e garantir que o dados do utilizador não são alterados.

Abstract

This dissertation consists on the idealization and implementation of Update Management Module for IPBrick Software. IPBrick is a company that providades professional solution through Unified Communication over IP and this solutions are deployed through their operative system and applications, like CAFE and iPortalDoc, that are a enterprise social network and a document management system, respectively. The Company has inserted itself in the Cloud world in the last years and from this insertion as resulted the necessity of creating a module that allowed their users update the sistem and its applications with the simples click of a button or with the definitions that guarantee an automated update. The module should adapt to the definitions that the user imposes and has to agree with the limitations that the software has so far and also the limitations that the system from which it's built from has, Linux Debian.

The module objectives and motivation are easilly explained with the use of a simple analogy. Lets think in a car of a random brandom that has a specific series and a specific volume and that although there a lot of features that are appreciated in the car, it would amazing if it was possible to maintain them but also move up on the series or maintain the series and move up on the volume. IPBrick intends to give an answer on how to do this by creating a shop where the car enters some parts are changed but the features that were so attractive and needed are maintained without any modification.

In conclusion, this implementation divides the module in two big parts: minor and major updates. Where the second isn't completely functional and still needs some work and the first, the SRRDR, allows the instalation of a new version of the OS with a simple reboot. The process before rebooting can be done in the background and doesn't need total service stop. The algorithm takes advantage of the free space by restructuring the partition scheme and then junctioning all the free space together in order to form a new partition to install the new operative system wihtout touching the user's data.

Agradecimentos

Em primeiro lugar, na minha mais sincera opinião, é necessário agradecer à Faculdade de Engenharia da Universidade do Porto (FEUP) e aos elementos que compõe a comunidade escolar, desde os estudantes até aos administradores passando pelos professores e funcionários. Independentemente de concordar com todas as medidas tomadas seja pelos elementos da direção ou pelos docentes a verdade é que esta ilustre faculdade foi a casa dos meus estudos durante 5 anos que culminam na escrita desta dissertação e a verdade é que sem esses 5 anos eu não poderia ser a pessoa que sou hoje nem o profissional que acredito que me vou tornar.

Em segundo lugar, é quase um dever agradecer aos meus orientadores, Engenheiro Miguel Ramalhão e Professor Ricardo Morla, pela paciência que tiveram face à minha desorganização muitas vezes e tendência para fazer entregas no limite dos prazos, embora as duas questões não tenham propriamente relação. Ao Eng. Miguel em particular por dificultar-me o trabalho, principalmente no que toca aos requisitos necessários, mas que me desafiou o suficiente para que eu continuasse a trabalhar para conseguir desenvolver o máximo que me fosse possível no tempo disponível num projeto que em grande parte do tempo passou por descobrir coisas novas e formas de dar a volta aos problemas. Ao Prof. Ricardo também por me dificultar a vida mas na parte da escrita, o que resultou num documento que tenta ser o mais completo e abrangente possível para que qualquer pessoa consiga entender mesmo sem ter os conhecimentos necessários para tal.

Em terceiro lugar, à IPBrick e todos os elementos que a constituem desde o CEO até ao programador ou comercial, desde os elementos alocados ao iPortalDoc até aos alocados ao IPBrick OS por me acolherem de forma tão calorenta, por me fazerem sentir parte integrante da empresa seja através de prendas para os colaboradores que acabam de ter filhos, corridas de 10 quilómetros, jantares de aniversário, histórias partilhadas à hora do almoço ou na pausa da tarde no Vício do Café, almoços de despedida ou qualquer outra forma de team building. Em particular, um grande obrigado ao Bruno Cochofel que me ensinou mais sobre Linux, programação e software do que qualquer outra pessoa no mundo e que teve a paciência necessária para lidar comigo quando as horas de sono são menos do que as horas de intervalo no trabalho e quando o raciocínio é lento ou inútil.

Em quarto lugar, à minha família que me apoia todos os dias e que me mantiveram em check durante toda a escrita. Obrigado pelo controlo excessivo fosse nas idas para a empresa ou nos horários cumpridos. Obrigado pelo interesse na empresa na qual estava a trabalhar e a estagiar. Obrigado pela preocupação em saber o que é que estava a fazer mesmo que não conseguissem propriamente entender com profundidade o assunto e em que é que este consistia. Obrigado pelas tentativas de ajuda mesmo que não fossem precisas ou possíveis.

Em quinto lugar, à minha namorada, Beatriz Barreira Leite, que foi a responsável muitas

vezes por garantir que eu cumpria os objetivos estabelecidos por mim próprio e sobretudo que me forçou muitas vezes a continuar a batalhar contra o projeto quando não parecia haver qualquer saída. Tenho ainda que lhe agradecer não só por me obrigar a explicar o meu trabalho da forma mais abstrata através da analogia mais improvável possível: "Imagina que tens um BMW série 3 com 2500 de cilindrada e que gostavas de passar para um série 5 sem ter de vender o carro e comprar um novo, queres simplesmente mudar meia dúzia de peças porque tens muito amor ao teu carro tal como é, pronto é isso!", mas também por ser o meu corretor das construções frásicas e da pontuação, sem ela esta dissertação seria mais aproximada a José Saramago e a textos chineses do que o que é suposto.

Em sexto lugar, à U.DREAM a primeira júnior social empresa que tem como principal missão mudar vidas e moldar consciências através da realização de sonhos a crianças num estado de saúde frágil e que me desafia todos os dias a ser uma pessoa melhor, um profissional melhor, a lutar pelo que acredito e a ser completamente maluco sem nunca ignorar as responsabilidades que tenho. Mas agradeço sobretudo aos seus elementos por me terem posto num departamento onde sou constantemente desafiado a pensar de perspetivas diferentes e a gerar novas ideias para criar mudanças positivas e melhoramentos no mundo e na minha vida. Em particular por me obrigar ao maior multitask e à maior compartimentalização que já consegui fazer.

Em sétimo lugar, a todos os meus amigos a quem tive de explicar em que é que a minha tese se baseava, utilizando a mesma analogia que criei para a minha namorada, e que se preocuparam constantemente comigo e com o estado do desenvolvimento da tese onde até se propunham a tentar ajudar mesmo quando não havia nada que pudessem fazer a não ser voluntariar-se para encontrar abreviaturas que eu tivesse deixado perdidas e sem explicação pelo texto, como o Eduardo Lopes fez, ou então distrair-me do trabalho e obrigar-me a fazer pausas quando o cansaço e o raciocínio não dão para mais. Em particular agradeço também ao Pedro Nuno Vilhena, o meu companheiro de tudo o que é FEUP que muitas vezes foi o meu salvador e que me ajudou a manter-me no caminho para satisfazer os objetivos necessários, à Isabel Fragoso, por ser a minha salvadora do semestre anterior e em muitas coisas deste também e por me tirar as dúvidas mais ridículas e absurdas de sempre e por ser o ponto de referência e o banco de informação para quem precisasse, à Mafalda Mendes, por me relembrar o quão mau é ser irresponsável e o quão complicado este projeto consegue ser às vezes, à Carolina Cunha e Costa, por nunca duvidar que eu ia conseguir ser capaz de entregar o que tinha para entregar dentro dos prazos e por querer sempre interromper o meu trabalho com as visitas mensais que faz.

Em oitavo e último lugar, quero agradecer a todos que não estão aqui referenciados especificamente devido à grande variedade de fontes e falta de classificação e que de uma forma direta ou indireta contribuíram para que este documento, que tem um número considerado de páginas e uma importância enorme para o fim do meu percurso académico, fosse concretizado e redigido. Desde os senhores que fazem a manutenção das máquinas de café até aos funcionários dos cafés que me vendem coisas gordurosas e nada saudáveis a transbordar de açúcar que permitem que consiga continuar o trabalho a que me tinha proposto.

Francisco Oliveira

"The more I want to get something done,
the less I call it work"

Richard Bach

"Laziness may appear attractive,
but work gives satisfaction."

Anne Frank

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

AM – Ambiente Minimalista
AMD – Advanced Micro Devices
APT – Advanced Packaging Tool
ATA – Advanced Technology Attachment
ATAPI – Advanced Technology Attachment Packet Interface
BIOS – Basic Input/Output System
BOOTP – Bootstrap Protocol
CD-ROM – Compact Disk Read-Ony Memory
CLI – Command Line Interface
CSS – Cascading Style Sheets
DB – DataBase
DBMS – DataBase Management Sysytems
DHCP – Dynamic Host Management Systems
DoS – Denial of Service
DRAM – Dynamic Random Access Memory
DPKG – Debian GNU/Linux Package Manager
DVD-ROM – Digital Versatile Disc - Read-Only Memory
EOF – End Of File
EXT – Extended file system
FAT – File Allocation Table
GB – GigaByte
GNOME – GNU Network Object Model Environment
GNU – GNU is Not Unix
GPL – General Public License
GRUB – GRand Unified Bootloader
GUI – Graphical User Interface
HDD – Hard Drive Disk
HIG – Human Interface Guidelines
HL – Hard Link
HTML – HyperText Markup Language
HW – HardWare
IaaS – Infrastructure as a Service
IBM – International Business Machines
IDE – Integrated Drive Electronics
IRQ – Interrupt Request
IP – Internet Protocol
KB – KiloByte
KDE – K Desktop Environment
LI – Live Install
LM – Live Migration
LVM – Logical Volume Manager
MB – MegaByte
MD5SUM – Message Digest 5 Sum
MFT – Master File Table
MGAS – Módulo para Gestão de Atualizações de *Software*
NAND – Not-AND
NI – Network Install
NIC – Network Interface Controller
NTFS – New Techonlogy File System
OS – Operative System
PaaS – Platform as a Service
PATA – Parallel Advanced Techonology Attachment
PB – PetaByte
PC – Personal Computer
PDF – Portable Document Format
PHP – PHP: HyperText Preprocessor
PMS – Package Management System
RAID – Redundant Array of Independent Disks
RAM – Random Access Memory
RDBMS – Relational DBMS
SaaS – Software as a Service
SATA – Serial Advanced Techonology Attachment
SAS – Serial Attached SCSI
SCSI – Small Computer System Interface
SL – Soft Link
SMS – Short Message Service

SPAM – Sending and Posting Advertisement in Mass
SQL – Structured Query Language
SRRDR – Sincronizar, Reestruturar, Reescrever, Descompactar e Reiniciar
SSD – Solid State Drive
SSH – Secure SHell
SSL – Secure Sockets Layer
SW – SoftWare
TB – TeraByte
TCP – Tranmission Control Protocol
TFTP – Trivial File Transfer Protocol
UDMA – Ultra Direct Memory Access
USB – Universal Serial Bus
VDI – Virtual Disk Image
VHD – Virtual Hard Disk
VM – Virtual Machine
VMM – Virtual Machine Monitor
VPN – Virtual Private Network
XML – eXtensible Markup Language
WWW – *World Wide Web*

Capítulo 1

Introdução

A *IPBrick* tem como um dos compromissos com o *software* que desenvolve servir o melhor possível os seus cliente através de um *software open-source*. Para a Empresa servir os seus utilizadores passa muito sobre permitir que estes tenham as definições ao seu gosto e que o sistema operativo para além de intuitivo seja fácil de usar e se comporte conforme as necessidades do cliente e as suas preferência de comportamento. O interesse do estudante aparece exatamente nesta crença. A crença de que o poder de decisão deve estar do lado do utilizador, que a máquina e o sistema devem moldar-se ao *user* a sua forma de comportamento ou às suas preferências e não o contrário como acontece geralmente, onde os clientes são obrigados a cingir-se ao que é imposto pelas empresas e pela forma como os sistemas são criados.

1.1 Contexto

A *IPBrick SA* constitui uma empresa especializada em soluções empresariais *UCoIP*, ou *Unified Communications over IP*. Possui vários produtos disponíves desde *deployment* de servidores em *cloud* ou *on-premisses* através da configuração de *intranet*, possui também uma rede social orientado ao ambiente de trabalho e ao ambiente empresarial e também serviços de *thin client*, virtualização, *e-mail*, entre outros. Com o desenvolvimento dos últimos anos da tecnologia de *Cloud* surgiu a necessidade de migrar os serviços para este tipo de tecnologia e responder às exigências do mercado. Nesta resposta surgem questões às quais é preciso também responder de forma a poder corresponder às expectativas geradas à volta da empresa.

Neste momento as atualizações do *software IPBrick* são geridas por um administrador que tem como principal responsabilidade isso mesmo, atualizar a plataforma seguindo todos os requisitos necessários e tudo é feito de forma manual, algo que precisa de ser alterado.

1.2 Motivação

Um cliente *IPBrick* que pretende apenas usar o sistema operativo não é obrigado a ter conhecimentos técnicos ou a agir de uma certa forma para poder utiliza-lo, é suposto tal como foi dito

anteriormente o *software* adaptar-se à pessoa. Emergindo portanto a necessidade de tornar todo o processo automatizado de forma a que o sistema se torne mais completo, fácil de usar, automatizado, personalizado e sobretudo mais eficiente. Desta forma para além de ser refidelizar os clientes atuais fornecendo um *software* cada vez mais adaptado às suas necessidades e fácil de usar, atrai também novos clientes.

1.3 Objetivos

Ao perceber as motivações que existem e o problema em si, torna-se imprescindível dotar os servidores *IPBrick* da capacidade de se "auto atualizar"mediante os critérios do utilizador e independentemente de onde estejam localizados ou do tipo de cliente que vai utilizar o *software*. Nesse sentido, o desenvolvimento do Módulo para Gestão de Atualizações de *Software* deve contemplar os seguintes objetivos:

- Estudo sobre o *software*, principais falhas, funcionalidades e limitações que possam influenciar o desenvolvimento do Módulo;
- Estudo e pesquisa sobre as limitações que são impostas pelo *Linux Debian* que é o sistema operativo no qual é baseado o da *IPBrick*;
- Investigar a possibilidade de conseguir fazer uma atualização na *Cloud* sem paragem de serviços ao cliente;
- Comprovar e desenvolver a possibilidade fazer uma atualização de distribuição sem ser necessário custos extra e acresencentar *hardware* virtualizado ou físico;
- Deve permitir que o utilizador escolha de que forma são feitos os *downloads*, se os ficheiros devem ser mantidos no sistema e os moldes da instalação.
- Deve permitir que o utilizador possa agendar as atualizações;
- Devem permitir que o utilizador seja notificado de novas atualizações, da instalação de uma atualização ou então das duas opções ou ainda de nenhuma;
- Deve permitir que uma atualização seja feita ou não dependendo do tipo de serviços que irá interromper;
- Interface Gráfica:
  - Desenhar os protótipos que darão origem à interface gráfica com todas as suas funcionalidades;
  - Histórico - listar atualizações anteriormente realizadas e os respetivos descritivos;
  - Virtual Hard Disk (VHD) Novas Atualizações - listar novas atualizações disponíveis;

  - Configurações - utilizador escolhe se pretende ser informado quando existirem novas atualizações, se pretende que as instalações sejam sempre feitas automaticamente, os seus descritivos e o que foi alterado;
  - Agendamentos - agendar a instalação de um *update* ou aplicações *on-demand* especifico ou a desinstalação do mesmo ou de outros.

- Módulo:

  - Desenhar o algoritmo que irá permitir verificar as atualizações disponíveis;
  - Garantir a integridade do sistema operativo e a possibilidade de Virtual Hard Disk (VHD)recuperar as configurações anteriores;
  - Permitir o *deployment* do módulo tanto em clientes *on-premises* bem como em *cloud*;
  - Atualizações de Distribuição:
    * Escalável para o futuro;
    * Devem implicar o menor transtorno possível para o cliente;
    * Devem tentar ser implementadas sem ser necessário, do ponto de vista do cliente, recorrer a elementos externos (como *pen USB*,*CD-ROM*, *DVD-ROM*);
  - Atualizações de Aplicações, Segurança, Reparação de Erros:
    * Deverão ser automatizados tanto em *download* como instalação;
    * Deverá procurar-se formas de aumentar a eficiência dos *updates*.

1.4 Estrutura da Dissertação

Esta dissertação está estruturada e divida em cinco Capítulos e dois Anexos. Onde o Capítulo 1 - Introdução - já apresentado onde constam o contexto da tese, as motivações e os respetivos objetivos.

O Capítulo 2 - Estado da Arte - é caracterizado pela apresentação das tecnologias existentes e dos conceitos necessários saber e compreender para perceber o enquadramento e trabalho desenvolvido bem como as limitações impostas e requisitos funcionais. Este capítulo serve também para introduzir o leitor às ferramentas que poderão ser utilizadas no capítulo 4 bem como os mecanismos do *Linux* e pressupostos que deram origem às ideias do capítulo seguinte.

O Capítulo 3 - Módulo para Gestão de Atualizações de *Software* - constitui o culminar da pesquisa feita para o capítulo anterior, onde são expostas as ideias que surgiram e o porquê de serem viáveis ou de não o serem bem como as tecnologias que serão utilizadas no capítulo seguinte para testes, desenvolvimento e implementação.

O Capítulo 4 - Implementação e Testes - este capítulo é o resultante dos testes efetuados bem como da tentativa de implementação e das dificuldades que se teve durante o processo bem como a forma como foram ultrapassadas.

O Capítulo 5 - Juízos de Valor e Trabalho Futuro - pretende fazer uma auto-avaliação do trabalho desenvolvido, que parte por analisar o que resultou e o que não resultou e possíveis explicações que possam justificar um ou outro desfecho.

Os Anexos A e B são utilizados por questões de estética e de facilidade em ler este documento. O Anexo A contém tabelas que ocupariam uma página caso fossem utilizadas juntamente com o texto e o Anexo B contém Figuras ou que são demasiado grandes e ocupariam uma página ou que correspondem ao mesmo parágrafo, parágrafo esse que já possui duas imagens seguidas (foi definido um máximo de duas imagens por parágrafo de forma a que não tornasse o texto confuso).

Capítulo 2

Estado da Arte

2.1 *Linux Debian*

O *Debian* é um sistema operativo, *OS*, *freeware* e *Open-Source* disponível para quem o quiser utilizar. Seja para fazer modificações, de forma a criar um *OS* em cima deste, ou, simplesmente, para usar como sistema operativo *standard*. Foi desenvolvido por um grupo de pessoas que pretende "abolir"a inflação exagerada dos produtos de *software* com o simples objetivo de disponibilizar para outras pessoas um sistema cada vez mais credível, gratuito e orientado para o seu utilizador.

Este *OS*, tendo em conta certos parâmetros de acordo com a *GNU General Public License*, permite que tudo, ou praticamente tudo, seja modificado ou que se acrescentem outras "peças"e por esta e outras razões, ter uma performance maior e permitir facilmente acesso sem recorrer a uma interface gráfica, que é largamente utilizado no desenvolvimento de novas ferramentas e na gestão de servidores de *Cloud*.

2.1.1 Discos e Controladores

O tipo de controlador utilizado nos discos rígidos que contêm os dados do sistema dá origem a nomenclaturas de partições diferentes, o que por consequência resulta em preocupações diferentes principalmente a nível da instalação do *OS*. Deste ponto de vista torna-se importante referenciar que tipo de discos é que se podem encontrar mas também de que forma é que isso influencia os sistemas que têm o *Linux* como base.

2.1.1.1 *Hard Disk Drive (HDD)*

Um *HDD* é um dispositivo cuja principal função passa por guardar e extrair os dados guardados. Isto é conseguido através da utilização de discos magnéticos juntamente com um "braço", as alterações magnéticas constituem "0's"e "1's"e, mesmo quando a energia não está disponível, o estado magnético mantém-se, conseguindo-se assim manter os dados.

O acesso à informação é feito de uma forma aleatória e não apenas de forma sequencial. O acesso pode ser bloqueado ou ser dificultado porque o disco está danificado. Este só é considerado danificado a partir do momento que acontece um *head crash* ou seja que o "braço", ou "agulha", arranha o disco e retira a película magnética que garante a integridade dos dados. O mau funcionamento da "agulha"pode acontecer devido a uma falha elétrica repentina, choques físicos ou então devido à alteração da pressão do ar uma vez que este precisa de ser constante para se poder introduzir as modificações e os dados no disco.

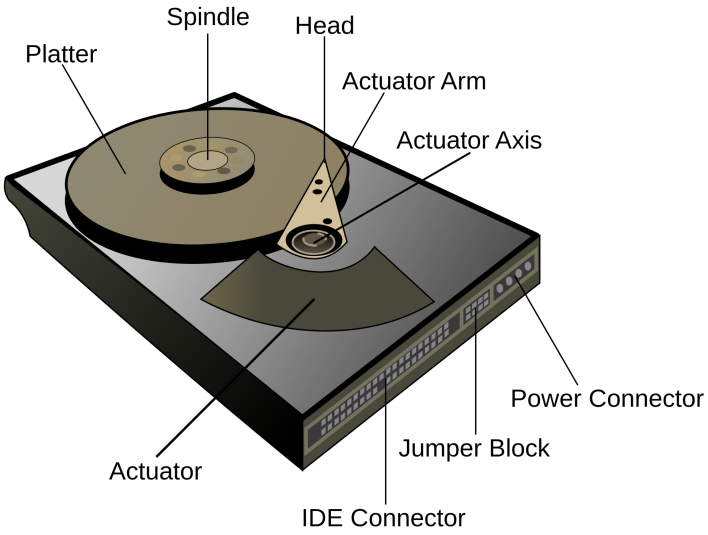

Figura 2.1: Componentes de um *HDD*.

2.1.1.2 *Virtual Disk Image (VDI)* e *Virtual Hard Disk (VHD)*

VDI é um formato de ficheiros utilizado maioritariamente no *software VirtualBox* na criação de máquinas virtuais onde a imagem funciona como disco físico. A imagem reside no sistema principal sob a forma de ficheiro. Quando o sistema secundário necessita de guardar informação ou inicia uma operação *I/O* a *VirtualBox* força o *bypass* e em vez de escrever diretamente para o disco físico escreve para o ficheiro, para o *OS* "convidado"e é como se estivesse a escrever para um disco, *standard* [11, 12].

Uma das grandes vantagens que os ficheiros do tipo *VDI* têm é o facto de ser possível expandir o ficheiro quando for necessário, contrariamente a um disco físico convencional [13].

VHD é também um formato de ficheiros utilizado na virtualização de discos para suporte de máquinas virtuais da *Microsoft* [14, 15]. Os ficheiros do formato *VHD* tal como os do *VDI* oferecem um número de vantagens importantes:

- Fáceis de Implementar - É possível utilizar um *VHD* com configurações predefinidas.
- Isolamento entre utilizadores - Cada utilizador tem o seu ficheiro e não consegue interagir com os outros diretamente.

- *Backup-and-Restore* - Quaisquer mudanças feitas, vírus ou *bugs* que existam são facilmente revertidos.
- Modificação *Offline* - É permitido aceder aos dados existentes no ficheiros sem ser necessário iniciar a *VM*.

Apesar destas vantagens os ficheiros *VHD* têm limitações, nomeadamente uma capacidade máxima de cerca de *2 TB* quando são utilizados controladores *RAID* e *SCSI* e uma capacidade máxima de *127 GB* para controladores virtuais *IDE*[13].

2.1.1.3 *Parallel Advanced Technology Attachment (IDE/ ATA/ ATAPI/ PATA)*

Inicialmente denominava-se *Integrated Drive Electronics (IDE)* ou *ATA-1* e obteve este primeiro nome devido ao facto de compreender o cabo e a interface integrados na *drive*.

Mais tarde surge a designação *Advanced Technology Attachment Packet Interface (ATAPI)* que pretende dar, de certa forma, a linguagem *SCSI* aos controladores *ATA* permitindo que estes sejam aplicados a mais tipos de discos sem terem de ser necessariamente *HDD*, como por exemplo: *CD-ROM*, *DVD-ROM*, etc.

Os controladores designados *PATA* não são mais do que o nome final dado ao desenvolvimento do *IDE/ATA* que embora descontinuado, ainda é utilizado atualmente em alguns computadores com *OS* igualmente descontinuado. O *Linux Debian Etch*, por exemplo, trata-se de um dos sistemas que se encontra descontinuado e cuja identificação dos discos é feita sob derivados de */dev/hda*. A última versão lançada refere-se ao *ATA/ATAPI-7*[1] e coincide onde começam a ser utilizados princípios de *SATA*, sendo capaz de atingir cerca de *133 MB/s*.

2.1.1.4 *Serial Advanced Technology Attachment (SATA)*

SATA é o controlador que utiliza sinais elétricos de alta velocidade para transferir dados e ligar dispositivos de armazenamento a um computador, como um ou mais *HDD*, leitores de *CD/DVD* e um ou mais *SSD*. Contrariamente ao *PATA*, este consegue oferecer mais velocidade, Tabela 2.1, e mais robustez com apenas sete *pin* pelo uso de um *serial bus* de alta velocidade, enquanto que o anterior necessitava de quarenta/oitenta *pin* [13].

Estes sete *pin* têm todos uma função específica: três são *Ground*, dois são de envio de dados (*A+/A-*) e os últimos dois são de receção (*B+/B-*). Apesar da utilização de um cabo elétrico de alta velocidade o efeito do ruído é evitado através da utilização de um sinal diferenciado o que permite isolar melhor os canais dedicados que o *SATA* tem para fazer as transferências. Isto constitui uma melhoria em relação ao *PATA* que apenas utiliza um sinal [16].

Cálculo das velocidades de transferência de dados:

1500MHz (embedded clock speed) x 1 (bit per clock)
x 80% (for 8b10b encoding) / 8 (bits per byte)
= 150 Mbytes/sec

[1] *UDMA/133 - Ultra Direct Memory* Access - Designação alternativa do controlador *ATA/ATAPI-7*.

SATA vs ATA/ATAPI	
Tipo e Versão do Controlador	Transferência de Dados
ATA/ATAPI-5	*66 MB/s*
ATA/ATAPI-6	*100 MB/s*
ATA/ATAPI-7	*133 MB/s*
SATA 1.0	*150 MB/s*
SATA 2.0	*300 MB/s*
SATA 3.0	*600 MB/s*

Tabela 2.1: Comparação entre *PATA* e *SATA* [6].

2.1.1.5 *Small Computer System Interface (SCSI)*

Neste tipo de controlador o cérebro da operação trata-se do adaptador ou controlador. Este elemento funciona como um "computador muito pequeno"e controla os dispositivos e o fluxo de informação. O adaptador é ainda responsável por receber a informação do sistema e os recursos necessários para as operações (*IRQ*[2] como endereços *I/O*) evitando portanto a necessidade de atribuir um elemento de cada um destes recursos a cada um dos dispositivos, sejam estes internos ou externos [17, p. 225–227].

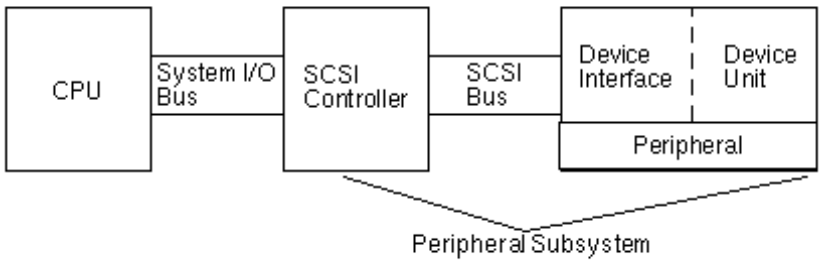

Figura 2.2: Esquema de funcionamento do adaptador *SCSI* [1].

2.1.1.6 *Serial Attached SCSI (SAS)*

Os protocolos *Parallel SCSI, PATA* e *SATA* têm certas limitações devido às tecnologias que utilizam, *buses*[18, 19], o que os torna pouco eficazes ao nível industrial. Estas limitações podem ser tanto no número de dispositivos que podem estar ligados como na velocidade de comunicação, a Tabela 2.2 apresenta alguns exemplos das limitações e faz a comparação com o *SAS*.

[2] *IRQ - Interrupt Request* - é a forma que o *hardware* pode requisitar tempo de computação. Por outras palavras, o processador executa um programa especial, denominado de *Interrupt Handler*, e depois volta à execução normal.

Limitações			
Controlador	Endereços	Tx Transmissão	Tam. Max. Cabo
Parallel SCSI	16	320 *MB/s*	12 m
FC[3] *SCSI*	126	*320 MB/s*	50 km
PATA	2	133 *MB/s*[4]	0.46 m (46 cm 18 *inch*)
SATA	15[5]	600 *MB/s*	1 m
SAS	128[6]	600 *MB/s*	10 m

Tabela 2.2: Comparação entre os controladores [7].

No fundo, o *SAS* aparece como substituto oferecendo melhorias na rapidez da comunicação e uma configuração mais simples combinando a velocidade de comunicação e transmissão do *SATA*, através da implementação dos cabos *point-to-point SATA* e portanto atingindo as mesmas taxas de transferência que estes possuem, com o endereçamento e inteligência do *SCSI*[7].

2.1.1.7 *Redundant Array of Independent Disks (RAID)*

Esta tecnologia foi criada para colmatar certos desejos que grande parte dos utilizadores de computadores hoje em dia têm. Nomeadamente o desejo de ter mais espaço, melhor performance e que o disco desse mais segurança, ou seja, que não fosse necessário estar constantemente preocupado com a possível perda de dados na falha de um disco.

A solução consiste na utilização de um vetor de discos independentes, ou *RAID*, que constituem em conjunto uma unidade lógica maior, mais rápida e mais segura.

Do ponto de vista externo, do utilizador, parece tudo relativamente simples, trata-se de um conjunto de blocos onde se fazem escritas e leituras, no entanto, ao nível interno torna-se bastante mais complexo. Para se formar um *RAID* são necessários vários discos e memória *RAM* mas também um ou mais processadores cuja a única função é otimizar a gestão do *hardware*.

A utilização de vários discos em paralelo oferece uma série de vantagens:

- Performance - Aumenta com a diminuição do tempo de dos acessos *I/O*. A informação estando partilhada pelos respetivos discos permite que estes façam leituras ou escritas em simultâneo em vários discos ao mesmo tempo logo aumentando a quantidade de informação processada por unidade de tempo.

- Capacidade - Aumentando o número de discos, aumenta-se a capacidade também.

- Confiança - Normalmente espalhar os dados por vários discos poderia correr mal caso um deles falhasse. No caso do *RAID* através de técnicas próprias e com uma certa quantidade de redundância permite que mesmo que se perca um disco completamente o sistema continue a funcionar sem qualquer problema.

- Transparência - Como o *RAID* aparece para o sistema operativo como um disco normal permite que qualquer disco seja alterado para um *RAID* sem interferir com o funcionamento do sistema ou sem ser necessário preocupações com compatibilidade.

Sempre que o sistema operativo pede um acesso lógico isso implica que seja calculado internamente que disco ou discos precisam de ser acedidos para realizar o pedido como tal para cada acesso lógico são necessários pelo menos dois acessos físicos internamente. Para além desta sobrecarga nos acessos físicos existe também um "senão"na capacidade. A quantidade ganha é diretamente proporcional ao número de discos utilizados mesmo aplicando redundância, no entanto, ao aplica-la para aumentar a tolerância a falhas e a capacidade de recuperação dos discos implica perder espaço. A quantidade de espaço perdido pode ser calculado através da formula:

$$UsefulCapacity = N * B[(k-1):k)]^{7} \tag{2.1}$$

Se o *RAID* é equipado com a capacidade de detetar e recuperar de falhas de discos é necessário que se perceba que tipo de falhas ocorrem e criar um modelo sobre como reagir a essas falhas. O modelo mais simples designa-se de *fail-stop* e deve ser levado literalmente, ou seja, se ocorrer uma falha o sistema interno assume imediatamente como se o disco estivesse inutilizado. Na realidade, acontecem mais vezes falhas de apenas uma secção ou um único bloco e para lidar com estas existem formas mais complexas [20, Capitulo 38].

Capacidade, tolerância a falhas, mecanismos de redundância, custo e performance dependem única e exclusivamente do que o utilizador pretende retirar da implementação. A Tabela 2.3, permite estabelecer algumas diferenças entre os diferentes níveis.

Custos e Ganhos dos Diferentes *RAID*					
Níveis	RAID 0	RAID 1	RAID 5	RAID 6	RAID 10
Min. Discos	2	2	3	4	4
Tolerância	None	1	1	2	1
Perda Cap.	None	50%	1	2	50%
Vel. Leitura[8]	N	N	N-1	N-2	N
Vel. Escrita[8]	N/2	1	1	1	N/2
Custo	Baixo	Alto	Alto	Muito Alto	Alto

Tabela 2.3: Níveis *RAID*

2.1.1.8 Influência dos Controladores e dos Discos

Convencionou-se que no diretório */dev/* são montadas tanto os discos como as respetivas partições de cada disco, sendo estas partições denominadas *hda, hdb* e *hdc* no caso de ser um controlador *IDE* ou *ATA* e *sda, sdb* e *sdc* no caso de serem todos os restantes (*SATA, SCSI, SAS* ou *RAID*) [21].

[7] N = nº de discos; B = nº de blocos; k = nº de cópias por bloco em diferentes discos

Após atribuídos os nomes aquando do *mount* dos respetivos discos a numeração das partições entra, como por exemplo: *hda1, hda2, sda, hdb1, hdb2, hdc*.

2.1.2 Núcleo do *Linux*

O *core* do *Linux* não é mais do que o seu *kernel* onde constam todas as funções base para este fazer operações, todas estas incluídas nos seus módulos. As ferramentas descritas de seguida são ferramentas que existem numa instalação básica porque permitem tornar a organização mais fácil e intuitiva. Para fazer testes e processos que envolvam mexer em zonas críticas permitem também mais segurança e isolamento através do *chroot*.

2.1.2.1 *Kernel*

A grande versatilidade apresentada pelo sistema advém, principalmente, deste ter um *kernel* que apresenta uma estrutura modular e este ser *loadable* ou, dito de outra forma, permite que sejam inseridos e retirados blocos *on-the-demand*, mesmo durante o funcionamento sem ser necessário reiniciar o sistema.

```
Module                  Size  Used by
linear                 12735  0
autofs4                27628  1
ipt_MASQUERADE         12594  1
xt_state               12503  4
xt_tcpudp              12570  31
iptable_filter         12536  1
iptable_mangle         12536  0
iptable_raw            12524  0
deflate                12551  0
zlib_deflate           25638  1 deflate
ctr                    12979  0
twofish_generic        16569  0
twofish_x86_64_3way    25167  0
twofish_x86_64         12541  1 twofish_x86_64_3way
twofish_common         20544  3 twofish_x86_64,twofish_x86_64_3way,twofish_gener
ic
camellia               29068  0
serpent                29015  0
blowfish_generic       12464  0
blowfish_x86_64        21201  0
blowfish_common        16487  2 blowfish_x86_64,blowfish_generic
cast5                  24829  0
des_generic            20851  0
```

Figura 2.3: Lista de Módulos de uma *IPBRick v6.1*.

Para ter acesso aos blocos que estão instalados no sistema basta abrir a linha de comandos e introduzir *lsmod*, para ter acesso a informação especifica de cada módulo utiliza-se o comando *modinfo* seguido do nome. Para inserir ou retirar um módulo utiliza-se, *insmod/rmmod*, respetivamente. Nas imagens seguintes aparecem alguns exemplos desta utilização:

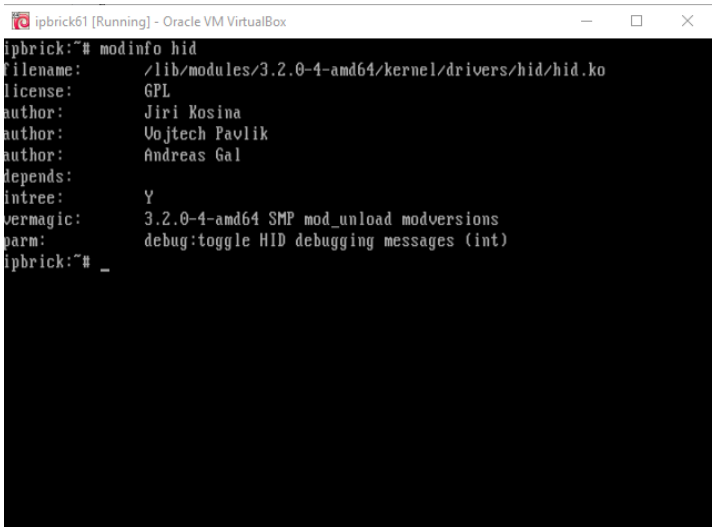

Figura 2.4: Listagem das informações de um módulo, neste caso do módulo *hid*.
9

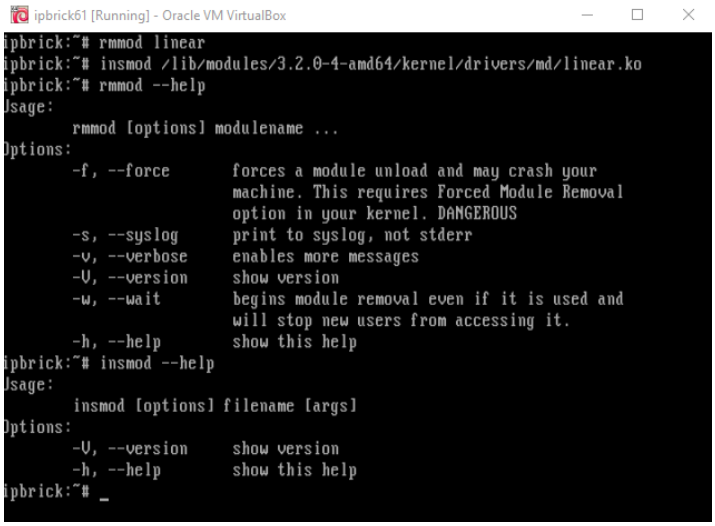

Figura 2.5: Inserir e Remover módulos.

2.1.2.2 *GRUB 2* e *GRUB Legacy*

GRUB é o acrónimo para *GRand Unified Bootloader* e como o próprio nome indica, *bootloader*, trata-se do primeiro programa a entrar em funcionamento quando um computador é ligado. Este programa muito poderoso é responsável por carregar e transferir o controlo para o *kernel* do sistema operativo que por sua vez vai iniciar o resto dos serviços necessários.

Foi descrito anteriormente que o *GRUB* era muito poderoso e efetivamente o é porque é extremamente flexível, ou seja, permite carregar o *kernel* de qualquer sistema operativo, seja *freeware*

ou proprietário (como o *Windows*). O *GRUB* entende os sistemas de ficheiros bem como os formatos de execução de um *kernel*. Isto resulta em ser possível usufruir de qualquer sistema operativo, no caso dos proprietários é apenas necessário executar com *chain-loading*[10]

O *GRUB 2* é uma evolução do *GRUB Legacy* e embora seja derivado deste segundo tem bastantes *features* adicionais:

- *grub.cfg* - Para além do ficheiro com as respetivas configurações ter alterado a sua nomenclatura (anteriormente *menu.lst*) passou a ser gerado automaticamente pela rotina *grub2-mkconfig*;
- Os números das partições passaram a começar com "1"e não com "0";
- Permite variáveis, condicionais e ciclos;
- Armazenamento mínimo é possível em certas configurações;
- Consegue encontrar *kernels* de forma mais consistente em sistemas com múltiplos discos e também consegue encontrar sistemas de ficheiros através dos seus *UUIDs*;
- Lê ficheiros diretamente de dispositivos *LVM* e *RAID* (ver Secção 2.1.1);
- Interfaces gráficas para o menu e para o terminal
- Faz o carregamento de módulos dinamicamente o que permite *kernels* mais pequenos (min.: *1 MB*).

A instalação desta ferramenta é feita nos sistemas *Linux* através do comando *grub-install /dev/ficheirododisco* onde se pode aplicar a opção *–boot-directory* caso se pretenda instalar numa pasta que não seja a predefinida (*/boot*).

2.1.2.3 Bash - Bourne-Again SHell

A linha de comandos do *Linux Debian* é uma das ferramentas mais importantes neste sistema operativo uma vez que é através desta que se consegue chegar ao mais baixo nível possível, o *kernel*. Tendo em conta a sua importância e a necessidade da sua utilização em ambientes de desenvolvimento baseados no sistema em questão é importante falar sobre o funcionamento e de que forma a *shell*[22] ou a *bash* influenciam o quão determinante esta ferramenta é.

Linha de comandos, frequentemente designada por *Bash*, ou *Shell*. Ou seja, ou um macro processador se se preferir, que executa comandos, por outras palavras, recebe texto e símbolos do utilizador com uma série de opções onde, após reconhecidas as palavras-chave e os respetivos símbolos, estes são transformados em expressões maiores que o computador consiga perceber para executar uma determinada rotina que vai originar o resultado pretendido.

[10] Operação que permite carregar um *kernel* que supostamente não seria suportado através do carregamento de um outro *bootloader*.

A linha de comandos pode ser usada de duas formas: Interativa ou Passiva [2, p. 1–2]. Interativamente onde o utilizador introduz comandos e estes são lidos e executados. Passivamente na leitura de um ficheiro com comandos, denominado de *shell script* e que aparece exemplificado na Figura 2.6, uma vez que a *shell* permite combinar comandos num ficheiro e este torna-se um comando também.

```
1
2  #!/bin/bash
3  # Counting the number of lines in a list of files
4  # for loop over arguments
5
6  if [ $# -lt 1 ]
7  then
8    echo "Usage: $0 file ..."
9    exit 1
10 fi
11
12 echo "$0 counts the lines of code"
13 l=0
14 n=0
15 s=0
16 for f in $*
17 do
18     l=`wc -l $f | sed 's/^\([0-9]*\).*$/\1/'`
19     echo "$f: $l"
20         n=$[ $n + 1 ]
21         s=$[ $s + $l ]
22 done
23
24 echo "$n files in total, with $s lines in total"
25
```

Figura 2.6: *Shell Script* [2, p. 5–39].

É também através da *shell* que os comandos e toda a biblioteca *GNU* é disponibilizada. Como em todas as linguagens de programação também estes comandos podem ser utilizados de forma síncrona ou assíncrona, sendo que na primeira espera-se pelo terminar da execução de certos comandos para iniciar a execução dos próximos e na segunda executa-se sem a necessidade de esperar.

Outra das possibilidades que torna este intérprete de linguagem de comandos tão apelativo é o facto de permitir controlo das entradas e das saídas bem como do conteúdo dos comandos. Isto é, permite indicar a entrada a utilizar, onde vai ser guardada a saída, caso exista e ainda que tipo de conteúdo é obtido ou como é apresentado [2, p. 41–68].

2.1.3 Ferramentas Importantes

Ferramentas que são consideradas importantes por toda a comunidade *Linux* e que são largamente utilizadas quando se pretendem fazer testes ao sistema operativo e possíveis implementações de código.

2.1.3.1 Ambientes Minimalistas

Chroot é uma abreviatura de *Change Root* e como a própria tradução indica, representa uma "Mudança de Raiz". Este comando pretende alterar a raiz do sistema. Temporariamente é criada uma *jail* onde a raiz aparente da árvore de diretórios é trocada pelo processo "pai"a correr e os

respetivos processos "filho". Em termos práticos, o que isto significa é que introduz-se numa outra pasta um sistema minimalista que tem as ferramentas necessárias para os testes que se pretendem fazer e este ambiente fechado permite fazer alterações. É importante nunca prejudicar ou pôr em risco o funcionamento do *host*. Deve ser ainda ressalvado que esta mudança de raiz, embora crie uma "prisão"que impede a interação com o exterior, é outra das *features* que se torna crucial no desenvolvimento e evolução de outros sistemas construidos sobre o *Linux* uma vez que através da utilização de *Hard Links* é possível conseguir acesso aos ficheiros do sistema original e desta forma testar o que se pretende implementar ou até mesmo introduzir alterações ao sistema [23].

Para se poder utilizar o comando *chroot* é necessário reunir algumas condições, nomeadamente a existência do sistema minimalista com todos os *packages* pretendidos na pasta que vai ser utilizada para a mudança de raiz, sendo que todo este procedimento pode ser acompanhado através dos seguintes comandos:

1. Garante que a ferramenta para instalação do sistema minimalista - *debootstrap*[11] - está presente no sistema - *apt-get install binutils debootstrap*.
2. Cria o diretório para o sistema minimalista - *mkdir /mydir/mychrootdir*.
3. Instala o sistema minimalista - *debootstrap –arch i386 mysystem /mydir/mychrootdir replink*. [12]
4. Entra na "prisão- *chroot /mydir/mychrootdir*.

2.1.3.2 *Links*

Links são ficheiros especiais, ou "etiquetas", utilizados largamente por muitos utilizadores como resultados das suas propriedades. Permitem que dois ficheiros com nomes diferentes se refiram ou apontem para o mesmo ficheiro ou diretório e podem aparecer de duas formas: *hard link* e *soft link*.

Hard Links Tal como foi introduzido anteriormente, na Secção 2.1.3.1, os *Hard Links*, ou *HL*, são elementos que permitem o acesso ao exterior mesmo após uma mudança de raiz, isto é possível porque um *HL* trata-se de uma referencia do mesmo diretório com nomes diferentes ou de um apontador para o mesmo nó para onde aponta o apontador original. Pode dizer-se, por analogia a circuitos elétricos que um *hard link* se trata de um "caminho paralelo". Para um entendimento mais fácil a Figura 2.7 abaixo inclui um esquema do funcionamento dos *HL* [24, p. 49].

[11] *Debootstrap* é geralmente usado para criar um sistema *Debian* diferente dentro de um subdiretório de outro sistema já em funcionamento e sem recurso a um *CD* de instalação, simplesmente por ligação a um repositório.

[12] As opções utilizadas no comando são necessárias para a correcta instalação sendo que:

(a) *–arch i386* - Indica que a arquitectura é de trinta e dois *bits*.

(b) *mysystem* - Trata-se de que tipo de *Linux* minimalista se pretende instalar.

(c) */mydir/mychrootdir* - Indica o diretório onde vão ser guardados os ficheiros.

(d) *replink* - Indica o *link* do repositório onde o sistema deve ir buscar os ficheiros para a instalação.

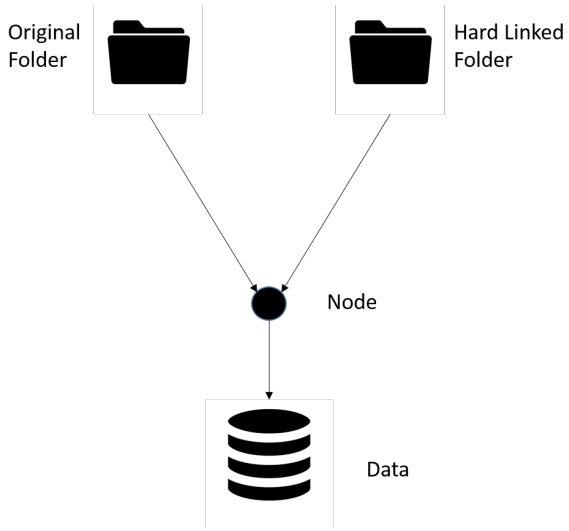

Figura 2.7: Esquema de funcionamento de um *Hard Link*.

Soft Links Para além da funcionalidade anterior existe uma outra, designada de *Soft Link* ou *SL* que também é gerada através do comando *ln* mas com a utilização da opção *-s* sendo que o que distingue um *HL* de um *SL* é o facto de o segundo funcionar como um apontador para o apontador da informação [24, p. 50], ou novamente por analogia, trata-se de um "caminho em série", tal como o esquema da Figura 2.8 demonstra:

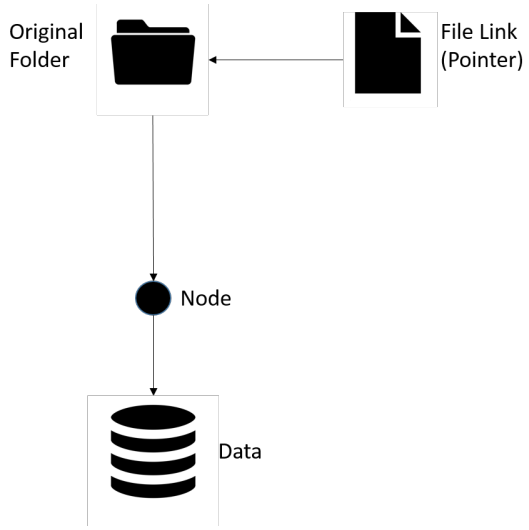

Figura 2.8: Esquema de funcionamento de um *Soft Link*.

2.1.3.3 *Shell Graphics*

Como todos os sistema operativos para conseguir ser utilizado por toda a gente, quer tenham conhecimentos de programação, *bash*, outras linguagens ou sem qualquer tipo de conhecimento

nesta área, necessita de ter uma interface intuitiva. Esta interface para além de intuitiva, tendo em conta que o ser humano depende imenso da visão, tem de ser também gráfica e é aqui que surgem o *KDE* e o *GNOME*.

Ambos constituem *GUIs*, *Graphic User Interfaces*, que estão disponíveis no *Debian*.

K Desktop Envrionment (KDE O *KDE Plasma*, cuja última versão é a 5.6.95, em utilização com a *Bash*, secção 2.1.2.3, formam uma *shell* com capacidades gráficas, ou seja, fornecem formas de manipular programas e aplicações através de uma *GUI*, Figura 2.9. Permitem ainda o que atualmente é algo extremamente intuitivo: fechar, abrir, redimensionar e/ou mover uma janela, aceder a ficheiros e outras funcionalidades [25].

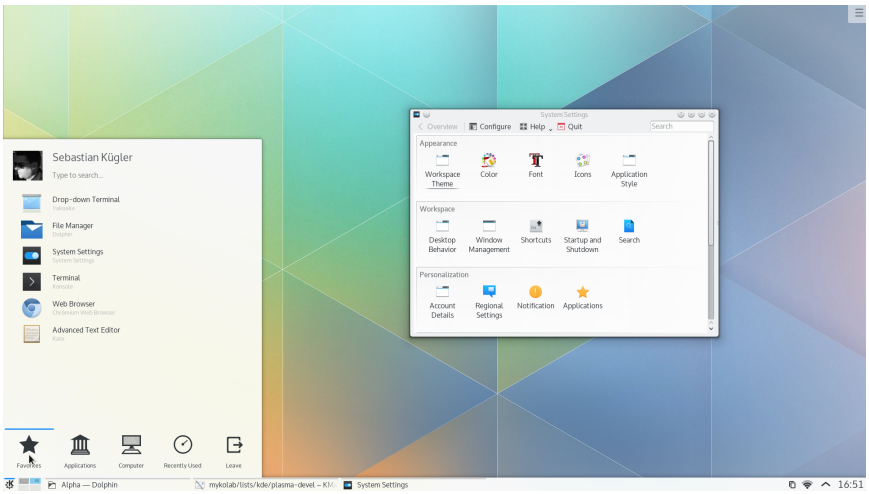

Figura 2.9: *Plasma Desktop Envrionment*

Esta *shell* gráfica é baseada no *KDE Frameworks*, cuja última versão é a cinco, que consiste, no fundo, num conjunto de bibliotecas e serviços que são precisos utilizar quando se quer usar aplicações *KDE*. Maioritariamente as bibliotecas são escritas em *C++* embora sejam aceites outras linguagens de programação através de tecnologias como:

- *Smoke* - Para o uso de *Ruby, Objective C#* e *PHP*
- *SIP* - Para o uso de *Python*
- *Kross* - Para o uso de *JavaScript, Falcon* e *Java*

GNU Network Object Model Environment (GNOME) O *GNOME* foi desenvolvido especificamente para o *Linux* embora possa ser utilizado noutros sistemas[26, p. 32–57]. O *GNOME Shell* é a interface gráfica do *GNOME Desktop Envrinonment* que permite a existência de uma *GUI*[27, p. 157–175] como a da Figura 2.10.

Desde a versão dois do *GNOME*, atualmente encontra-se na *v3.20.1*, que o foco é o pragmatismo e a produtividade. Nesse sentido, foram criadas as *HIG - Human Interface Guidelines* -

cujo objetivo é definir uma série de normas baseadas em estudos ergonómicos e cognitivos[28, p. 312–328][29] que se dedicam a perceber os processos mentais básicos como: perceção, memória, pensamento, resposta motora e etc [30, 31]. Com o objetivo de se conseguir[29]:

- *Design* a pensar no utilizador;
- Manter a ligação entre a aplicação e o mundo real;
- Aplicações consistentes;
- Manter as pessoas informadas;
- Simples, mas bonito;
- Personalização .

Figura 2.10: *GNOME Shell.*

As normas *HIG* são as que permitem que os *developers* nunca se esqueçam que a interface gráfica esteja sempre em harmonia e que é orientada para o utilizador. Devido a esta abordagem é possível estabelecer forças, fraquezas, ameaças e sobretudo oportunidades:

GNOME SWOT			
Forças	Fraquezas	Oportunidades	Ameaças
Suporte em várias linguagens.	Informação pouco atualizada e estruturada.	Estar disponível para soluções empresariais.	*Microsoft* detém 92% do mercado.
Recursos de Consultoria.	Peso da estabilidade e da inovação não se encontra balanceado.	Integração com a computação na *Cloud*.	O desenvolvimento *Mobile* é feito com pouca comunicação.
Fácil de usar e incorporar.	Valores do*GNOME* estão a ser mal representados no projeto.	Acolher serviços *WEB*	Perda de colaboradores devido à má reputação de uma das aplicações.
Grande suporte económico.	Pouco desenvolvimento de ferramentas escolares.	Melhorar a comunicação entre colaboradores.	Maior preocupação com desenvolvimento de *Apps* do que com o projeto.
Top grade Middleware	O modelo atual não permite mudanças radicais, está orientado para melhorias localizadas.	Encorajar empresas e investigadores a pensar fora da caixa.	Empresas que utilizam as *Apps* desenvolvidas não as melhoram, não consideram vantajoso.

Tabela 2.4: Análise *SWOT* do *GNOME* em 2010 [8].

2.1.4 Sistemas de Ficheiros

Um sistema de ficheiros utiliza-se para conseguir indexar dados, para conseguir recuperar depois de guardados, apagar, editar ou outra operação que implique a interação com informação guardada no disco. Sem um *filesystem* o que aconteceria seria que se teria uma pilha de informação da qual não se conseguiria obter conteúdo, uma vez que não se saberia onde começa e acaba cada parte do mesmo.

No sentido de evitar este problema e poder indexar o que se pretende guardar para depois consultar ou até obter por algum motivo, foram criados sistemas de ficheiros.

O *Linux* sendo um sistema operativo derivado de pessoas com bastantes gostos diversificados e com ideias diferentes precisava de se adaptar a essas pessoas também no que toca ao mapeamento dos ficheiros por oposição a outros concorrentes do mercado onde é imposto uma certa utilização ao cliente, como por exemplo: o *Windows*. Que tem como predefinição o *NFTS* ou o *FAT* [20, 32].

2.1.4.1 *File Allocation Table (FAT)*

Os ficheiros neste sistema são organizados por *clusters* de dados onde o apontador do *cluster* está nos ficheiros da Tabela de Ficheiros Alocados criada aquando da formatação do disco ou da partição do disco[33, p 125–128].

Cada célula da tabela para além de conter o indicador para o próximo *cluster* pode também ter outro tipo de indicador:

- *0xFFFF - EOF - End of File*
- *0x0000* - Espaço Livre
- *0xFFF7 - Cluster* corrompido

FAT em conclusão é um sistema simples e robusto, apresenta uma boa eficiência para pequenas utilizações, mas não consegue competir com os *filesystems* modernos em termos de *performance*, escalabilidade e robustez.

2.1.4.2 *New Technology File System (NTFS)*

O *NTFS* é o sistema de ficheiros predefinido do *Windows*, que também é compatível com o *Linux* e que surgiu da necessidade da *Microsoft* ter um sistema confiável, simples, rápido, seguro e que pudesse ser utilizado pelo setor corporativo ou por *users* que não usam o computador repetidamente e de forma intensiva na modificação de ficheiros. Esta necessidade foi criada devido ao facto do *FAT* (Secção 2.1.4.1) não ser capaz de satisfazer este requisito.

Neste sistema de ficheiros tudo é considerado um ficheiro (com a exceção da partição *boot*) ou melhor uma *stream* de atributos[13], mesmo as estruturas que fazem a gestão das partições e que guardam as estatísticas e informações de controlo são também ficheiros especiais [32].

A ideia por trás do recurso a uma estrutura de ficheiros baseada em atributos, como é o *NTFS*, centra-se no funcionamento idêntico a uma Base de Dados (Ver Secção 2.6.2). Isto permite que o sistema operativo veja os ficheiros como objetos com certas características definidas o que otimiza a pesquisa destes e a sua alteração quando for necessário [32].

À semelhança do *FAT* os ficheiros, incluindo os de *metadata* são guardados usando um sistema de *cluster* (setores de 512 *bytes*). A razão para organizar tudo desta forma é por uma questão de eficiência, poupança de recursos e evitar a fragmentação. Se os setores fossem guardados individual e aleatoriamente num disco de grande dimensão seriam necessários muitos recursos para saber onde está a informação e esta poderia apresentar-se muito dispersa [32].

Mesmo organizando os ficheiros da forma descrita no parágrafo anterior é necessário haver um "centro de comando"ou alguma estrutura que saiba encaminhar o sistema operativo para o *cluster* que tem a informação que este quer. A estrutura responsável por isto trata-se da *Master File Table*, ou *MFT*, e funciona como uma base de dados relacional, como um índice do disco onde sempre que um ficheiro ou diretório são criados a *MFT* regista esse evento através de atributos[14] [32].

[13] Uma *stream* porque cada um destes ficheiros são linhas de atributos.

[14] Os atributos podem ser *residents* ou *non-residents*, portanto, fixos ou variáveis e são guardados na *MFT* ou numa das suas extensões, respetivamente

2.1.4.3 *Extended File System (EXT)*

O *filesystem* dos sistemas *UNIX* é o *EXT* e funciona através de *index nodes* ou *inodes* e de *links*. No fundo o que acontece é que a um ficheiro corresponde um nó e esse pode ter várias ligações a diversos pontos diferentes no sistema e qualquer um destes pontos tem acesso ao ficheiro e ao seu conteúdo, isto implica que mesmo eliminando um dos *links* o ficheiro continua acessível. Esta forma de funcionamento através do nó permite também que seja possível aceder, apagar ou editar um ficheiro noutra aplicação [34, 20].

Este sistema é caracterizado por ter diretórios onde cada um é uma tabela onde cada linha associa o nome do ficheiro com o número do nó. Nó esse que consiste no número, no tamanho do nome e do próprio nome. Para encontrar um ficheiro é necessário pesquisar de frente para trás na procura do nome associado, para diretórios de tamanho razoável é suficientemente eficiente. No que toca a diretórios grandes já não se verificava isso apenas com este funcionamento.

Muitos sistemas operativos usam *B-Tree*[35] que permite indexar diretórios e obter uma performance elevada mas esta forma de o fazer não combina com o *EXT*, que se tornaria o sistema de ficheiros predefinido no *Linux*. Para suprir esta lacuna foi desenvolvida uma outra forma de indexar os diretórios e aumentar a eficiência, denominada de *HTree* [15].

O *EXT4* é a última versão deste método de organização de discos que resultou de extensões ou melhorias impostas ao *EXT3* e portanto mantém como todos os outros a compatibilidade com versões anteriores e acrescenta-lhes uma série de características importantes em termos de *performance*, robustez e organização:

- Maior Capacidade de Indexação
- Mapeamento Melhorado - Foi aumentada a capacidade de mapeamento e reduzida a fragmentação, o que resulta num aumento da eficiência principalmente no tratamento de grandes blocos de dados.
- Mantém Compatibilidade - É possível montar *EXT2/EXT3* como *EXT4*. Desativando algumas funções é possível também montar *EXT4* como *EXT3* (*forward compatibility*).
- Pré-Alocação Persistente - Permite a alocação de espaço para um ficheiro antes deste ser criado.
- Atrasar Alocação[16] - Torna-se possível atrasar uma alocação de blocos de espaço livre enquanto a informação não é escrita para o disco diretamente (Muitas vezes antes de a informação ser guardada em disco é guardada em *cache*, no entanto o espaço é automaticamente alocado e fica em espera, sem ser necessário, enquanto a *cache* está a ser escrita).
- Mais Sub-diretórios - *EXT3* tinha um limite de cerca de trinta e dois mil que no *EXT4* não existe, dando origem a ilimitados sub-diretórios.

[15] A *HTree* consiste simplesmente numa *B-Tree* onde se aplica um algoritmo que gera *hashes* uma função *hash*, cujo objectivo é ajudar no mapeamento de informação [35].

[16] Esta*feature* pode levar de alguma forma a perda de informação quando acontecem falhas de eletricidade

- *Journal Checksum* - Cria-se uma "etiqueta"que garante que o *Journal*[17] não se encontra corrompido o que aumenta a confiança no processo e melhora a eficiência uma vez que não existem tantas falhas.
- Verificação Rápida de Ficheiros - Os blocos que não estejam alocados estão assinalados na tabela que contém os nós indexados e na altura da verificação dos ficheiros estes são ignorados para tornar a verificação mais rápida e eficiente.
- Alocar Múltiplos Blocos em simultâneo - Graças à utilização da *feature* que atrasa a alocação de espaço é possível manter os dados num *buffer* e alocar grupos de blocos conseguindo desta forma de seguida atribuir de forma mais eficaz os dados aos respetivos blocos ao mesmo tempo.
- Melhoria na Apresentação do Tempo - No *EXT4* o tempo passou a ser medido em nanosegundos.

Uma das outras razões que torna esta forma de acesso tão vantajosa é porque o descrito no primeiro parágrafo não acontece no *Windows*, por exemplo, onde automaticamente impede-se uma modificação do ficheiro quando este se encontra aberto numa outra aplicação[9, p. 73–76].

2.1.4.4 *FAT vs EXT vs NTFS*

Na Tabela 2.5 pode observar-se algumas diferenças entre os respetivos *filesystems* bem como alguns números que os diferem.

Sistemas de Ficheiros e Informações			
Filesystem	Tam. Max. Ficheiros	Tam. Max. Partições	Cache
FAT16	2 GiB	2 GiB	No
FAT32	4 GiB	8 TiB	No
NTFS	2 TiB	256 TiB	Yes
EXT2	2 TiB	32 TiB	No
EXT3	2 TiB	32 TiB	Yes
EXT4	16 TiB	1 EiB	Yes

Tabela 2.5: Comparação entre os diversos sistemas de ficheiros existentes e possíveis de utilizar no *Linux Debian*[9, p. 73–76].

2.1.5 Obter o *OS*

Existem várias formas de instalar ou de obter um sistema operativo, neste caso específico o *Linux Debian* numa máquina, ou mais comummente designado *Personal Computer*, *PC*, sendo principalmente as seguintes:

[17] Trata-se de uma forma de seguir as alterações que foram feitas antes destas serem implementadas no disco que constitui um *failsafe* no caso de haver algum problema externo. Evita também que na ocorrência de um problema se evite que os dados fiquem corrompidos.[36]

1. *Live Install or CD/DVD/USB*

2. *Network Instal*

3. *Cloud Deployment*

É de grande importância salientar que todas estas opções implicam, normalmente, o *overwrite* do disco, ou seja, na existência de outros dados, e após confirmação do *user*, estes serão apagados e substituídos por novos.

2.1.5.1 *Live Install*

Aquilo que se designa por *Live Install*, doravante *LI*, é a instalação através de um executável do sistema após a extração dos ficheiros necessários para a sua realização, contidos num arquivo *ISO*. Por outras palavras, qualquer utilizador pode descarregar o ficheiro compactado com toda a informação de um dos *websites* oficiais do *Debian*, extrair a informação necessária através de programas ou ferramentas embutidas para o efeito, ou como alternativa gravar o ficheiro descarregado diretamente num *CD, DVD* ou *USB*, para que posteriormente uma destas opções possa ser introduzida no *PC* e a instalação é iniciada automaticamente.

2.1.5.2 *Network Install*

Network Install, *NI* facilitando, como a sua tradução indica, corresponde a uma instalação pela rede, mais precisamente, através da Internet. *NI* baseia-se na configuração de dois servidores, *TFTP (Trivial File Transfer Protocol)* [18] e *DHCP (Dynamic Host Configuration Protocol)* [19] geralmente, embora se possa utilizar *BOOTP (Bootstrap Protocol)* [20] ou *RARP (Reverse Address Resolution Protocol)* [21] também.

2.1.5.3 *Cloud Deployment*

Cloud Deployment funciona de igual forma para todos os sistemas operativos, ou seja, uma máquina é lançada com as características pretendidas pelo *user* mediante a utilização que vai ser dada e assim que é iniciada a máquina através de uma *ISO*. Também se inicia o processo de instalação da mesma forma que seria se se tratasse de uma máquina física onde se introduzisse o *CD* ou o *DVD* de instalação. Mais à frente serão explicados os conceitos de *Cloud* e de virtualização.

[18] O *TFTP* é o protocolo responsável pela transferência de ficheiros de um servidor remoto para a máquina a ser utilizada[37]

[19] O *DHCP* é o protocolo responsável por estabelecer as configurações dos *sites* dinamicamente. Posto isto, o protocolo atribui um endereço temporariamente até deixar de ser necessário. [38]

[20] O *BOOTP* automaticamente atribui endereços a um equipamento dentro da rede através de um servidor de configuração. [39]

[21] O *RARP* permite a descoberta do endereço de *IP* através do *MAC Address*, por analogia, é o protocolo que permite saber o "nome"da máquina na camada da *Internet* sabendo o "nome"do *hardware*. [40]

2.2 Virtualização

Explicar o conceito de virtualização implica explicar que existem dois tipos: *Bare-Metal* ou tipo 1 e *Hosted* ou tipo 2. Cada um destes tem características específicas que vão ser frisadas de seguida, bem como vantagens e desvantagens associadas, sendo que esta primeira secção funcionará apenas como matéria introdutória para o tipo de serviços integrados que a *IPBrick* fornece num contexto de *Cloud* e de virtualização de forma a que seja de fácil compreensão os problemas gerados e a necessidade que existe pelo desenvolvimento deste projeto.

2.2.1 Virtualização Tipo 1

Este tipo de virtualização como o próprio significado em Inglês permite perceber, "Diretamente no Metal", tem apenas o *Virtual Machine Manager (VMM),* a separar a Máquina Virtual (*VM*), do *Hardware (HW)* ou Máquina Física. Este tipo de implementação apresenta duas desvantagens e duas grandes vantagens: complexidade de instalação e de configuração, grande desempenho ao nível do processamento e melhor utilização de periféricos de *I/O* [41, 42], respetivamente.

A Figura 2.11 ilustra o esquema da virtualização do tipo 1.

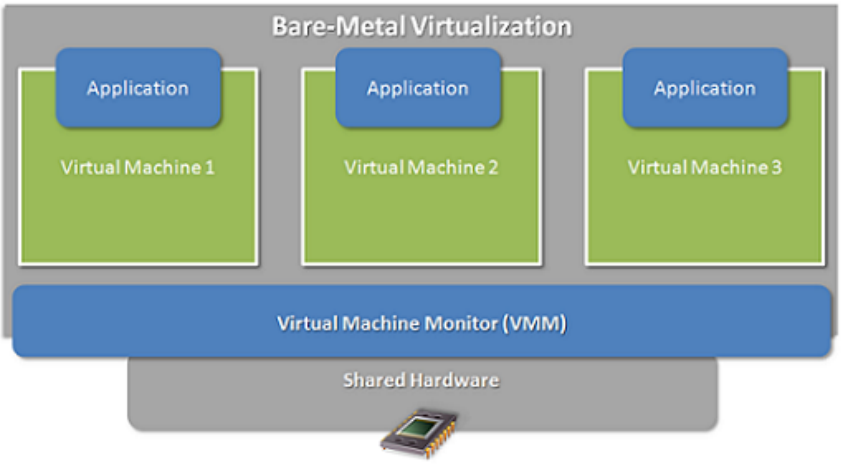

Figura 2.11: Esquema ilustrativo da Virtualização Tipo 1.

Existe justificação para se utilizar este tipo de virtualização, para além da melhoria de desempenho devido da inexistência de um sistema operativo, *OS*, como camada intermédia entre *VMM* e *HW*, permite um maior grau de liberdade e aplicações disponíveis, apesar de contemplar uma implementação mais complexa.

Uma das formas de utilizar este tipo de virtualização é através do *kernel* do *Hypervisor*, como o esquema da Figura 2.12 o demonstra.

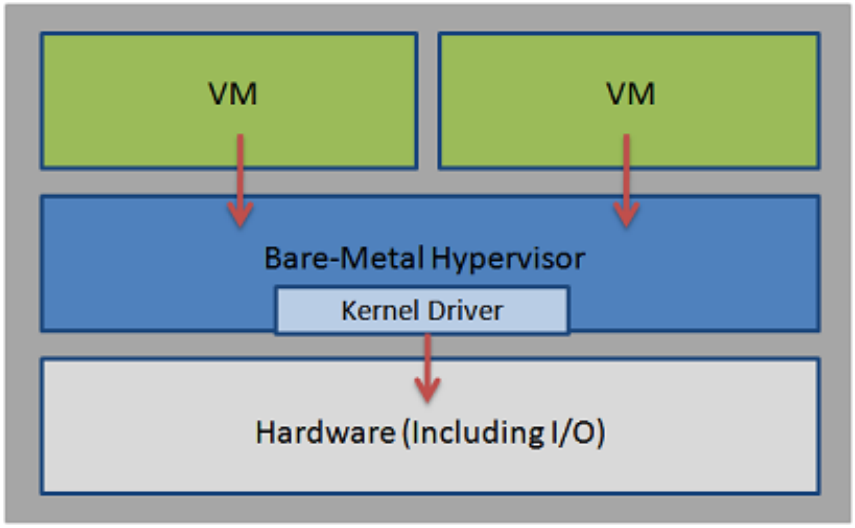

Figura 2.12: Virtualização Tipo 1 - *Hypervisor* como intermediário.

No esquema acima, Figura 2.12, como se pode ver pela representação de alto nível, existe um encadeamento de instruções. Primeiro para o *kernel* e só de seguida para o *HW*. Mesmo assim este sistema apresenta um certo *delay* comparativamente ao que seria o funcionamento de uma máquina física, uma vez que as instruções têm de se passar ao *Hypervisor* antes de serem executadas no *HW*.

O esquema seguinte, Figura 2.13, apresenta o que é a Virtualização Tipo 1 particionada que permite a redução do efeito de *delay* da camada do *VMM* e a comunicação direta com os elementos físicos através da alocação e reserva de recursos para aquele sistema virtual [41, 42].

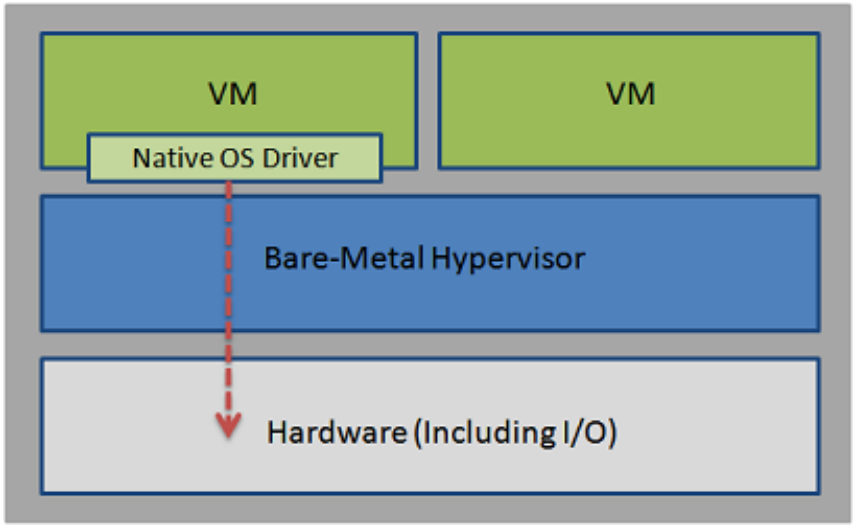

Figura 2.13: Virtualização Tipo 1 - *Partitioned Resources*.

2.2.1.1 Virtualização Completa

Na primeira instância e relativamente fácil de perceber, o sistema é completamente virtualizado, ou seja, o *hypervisor* cria uma "imagem"da máquina física e o sistema funciona sem conhecimento da estrutura virtual, Figura 2.14.

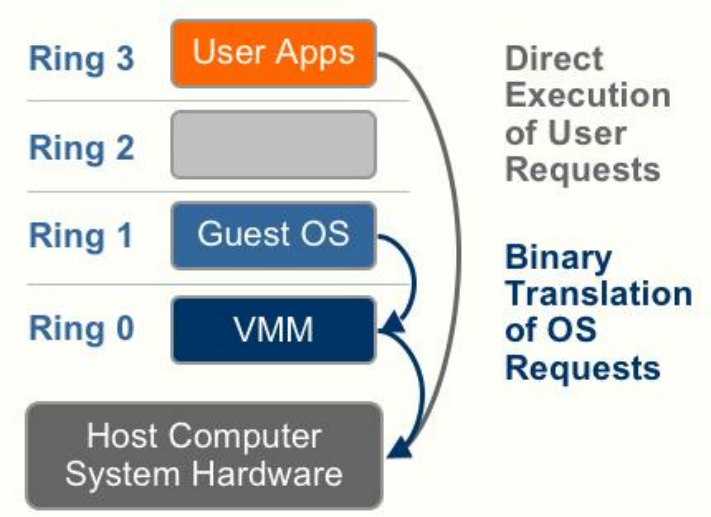

Figura 2.14: Virtualização Completa. [3]

2.2.1.2 Para-Virtualização

Na para-virtualização, Figura 2.15, e para esta ser implementada, é necessário existirem modificações no sistema operativo virtualizado. Continua a haver uma necessidade de "triagem"das instruções geradas pela virtualização mas estando o *OS* modificado torna-se mais compatível com o *VMM* e portanto evita-se a verificação de toda e qualquer instrução aumentando, consequentemente, o desempenho [41, 43, 42, 3].

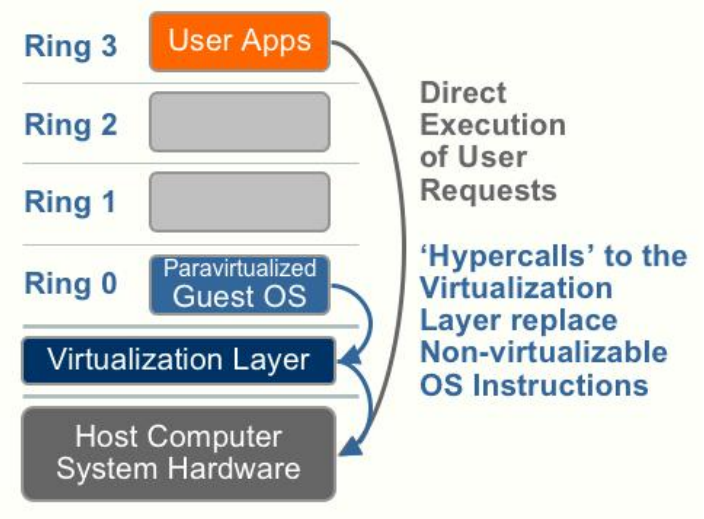

Figura 2.15: Para-virtualização[3]

2.2.1.3 Virtualização Assistida

Por último temos na Figura 2.16 algumas instruções e um esquema de como funciona a virtualização assistida por *hardware*. Esta virtualização é baseada em sistemas desenvolvidos pela *Intel* ou pela *AMD*. Este tipo é diferente da para-virtualização no sentido em que não modifica o *Guest OS* para poder interagir com o *Hypervisor* e com o *HW* em vez disso este sistema permite transferência mais rápida do controlo entre os anéis de virtualização e maior segurança na alocação de recursos de *I/O* [43, 42, 3].

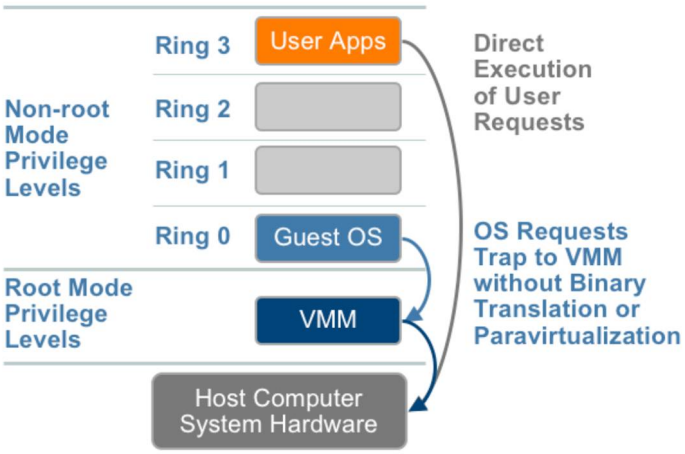

Figura 2.16: Virtualização Assista por *Hardware*[3]

2.2.2 Virtualização Tipo 2

O segundo tipo de virtualização, *Hosted* ou Tipo 2, consiste numa camada extra na *stack* de virtualização. Esta camada extra é apenas o sistema operativo que corre nas máquinas físicas que fornecem os recursos aos sistemas virtualizados, Figura 2.17.

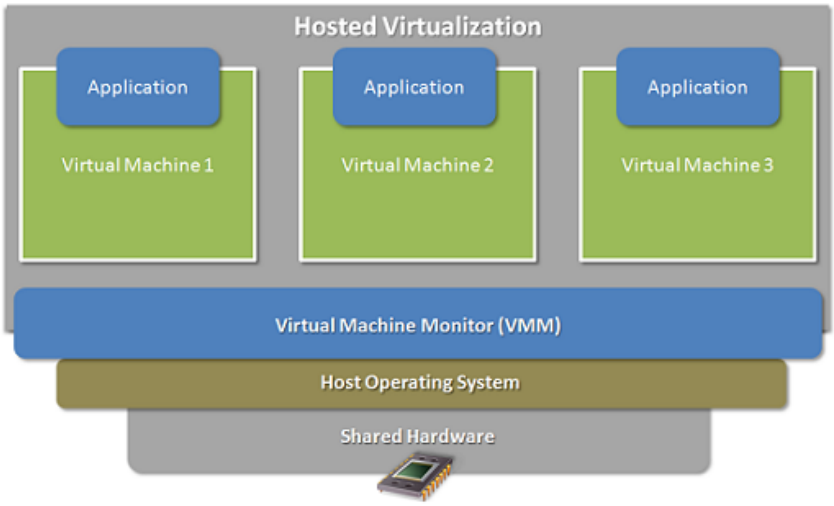

Figura 2.17: Esquema Ilustrativo da Virtualização Tipo 2.

A existência de mais uma camada implica que este tipo de virtualização tenha um desempenho menor do que o Tipo 1. A Figura 2.18 ilustra o caminho que é gerado para haver comunicação entre o sistema virtualizado e o *HW* para que se possa compreender melhor o porquê desta conclusão ser obtida.

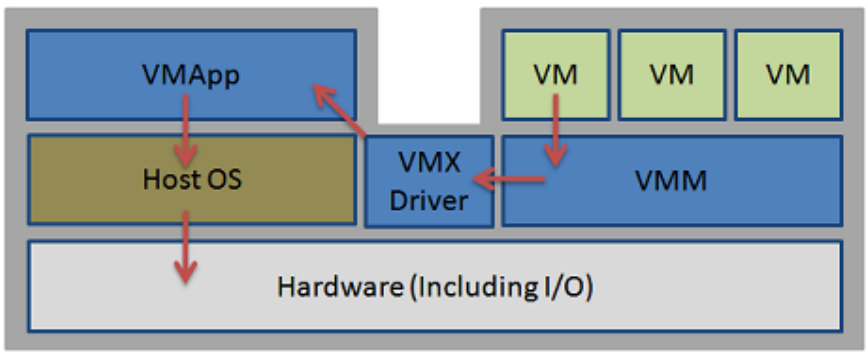

Figura 2.18: Virtualização Tipo 2 - *Communication Path*.

2.3 *Cloud*

Considera-se interessante e importante uma introdução ao conceito de *Cloud* (como é que funciona, para que serve e o porquê de ser utilizado esta tecnologia) mas também a algumas

características particulares que apresentam uma relevância maior para o problema em questão.

O conceito é definido a nível macroscópico como um aglomerado de computadores que vão ser utilizados para diversas funções. Esta tecnologia permite gerar uma camada virtualizada dependendo das necessidades de um consumidor (seja capacidade de processamento, disco ou capacidades gráficas), sendo que este apenas precisa de ter conhecimento de como utilizar esses mesmos recursos sem ter de se preocupar com a complexidade e/ou custos de manutenção, desenvolvimento, *backups*, etc[41].

2.3.1 Implementações

O consumidor não tem acesso ao funcionamento interno da nuvem e pode apenas escolher as características virtualizadas das máquinas que pretende ou o nível de abstração. Todas as implementações faladas abaixo podem ser verificadas esquematicamente na Figura 2.19.

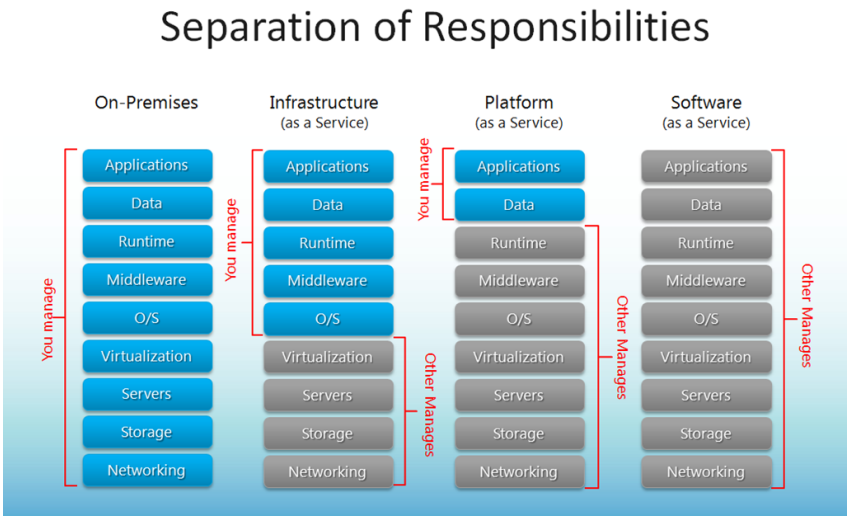

Figura 2.19: *Cloud Stack*.

2.3.1.1 *IaaS - Infrastructure as a Service*

Infraestrutura como um serviço, ou *IaaS*, Figura 2.19, é o nível mais baixo de abstração possível e é o único que permite a escolha de componentes como poder de processamento, memória em disco, memória *RAM* disponível, componentes de Rede, etc[44]. Neste tipo de implementação o cliente fica com uma camada virtualizada como um todo que pode ser utilizado para testes de um projeto que está a ser desenvolvido, evitando assim os custos adquirir, fazer a manutênção de servidores de teste, fornecido a outros consumidores através do desenvolvimento e criação de *software* que fica disponível como *SaaS* ou então pode ser desenvolvido a plataforma para outras pessoas desenvolverem as suas próprias aplicações e neste caso trata-se de *PaaS*.

2.3.1.2 *PaaS - Platform as a Service*

Plataforma como um serviço, ou *PaaS*, Figura 2.19, considera-se o nível médio de abstração uma vez que a nuvem é responsável por fornecer ao *user* um ambiente de desenvolvimento, isto é, um ambiente que permita o teste, desenvolvimento, o suporte e entrega a um consumidor final um produto *Web* para utilização sem necessidade de instalação ou *download*[41][44]. *PaaS* permite que se consiga criar novo *software* com tudo o que é necessário para o mesmo sem preocupações, ou seja, sem a necessidade de ter custos numa infraestrutura nem na configuração de uma plataforma ou em sistema de manutenção uma vez que tudo isso está assegurado pelo fornecedor.

2.3.1.3 *SaaS - Software as a Service*

Aplicações como um serviço, ou *SaaS*, Figura 2.19, é o nível mais alto de abstração, representando aquele que teria menos encargos tanto na parte do desenvolvimento como na parte do consumo. Isto significa que todas as camadas anteriormente descritas são geridas pelo *cloud provider* (por exemplo: *Amazon*, *Google*, *Microsoft*, ...) e o *end-user* não tem sequer de se preocupar com *hardware* ou com servidores nem mesmo com atualizações uma vez que é tudo feito internamente, sendo deste modo, apenas visível o serviço do ponto de vista exterior. No fundo, isto significa que *SaaS* na grande parte das vezes são *thin clients*[22], por outras palavras, uma empresa que pretenda lançar uma aplicação através de um fornecedor de *SaaS* que vai garantir que tudo está disponível para o consumidor final utilizar [41][44].

2.3.2 *Live Migration*

Live Migration, ou *LM* para ser mais simples, é um algoritmo ou dinâmica que tem como objetivo minimizar o tempo em que um serviço fornecido por uma nuvem, ou seja os sistemas virtualizados, estão *down* quando existe a necessidade de mudar de *VM* ou transferir o estado de memória para efeitos de balanceamento da carga do servidor e para consolidação de servidores[5]. Esta dinâmica tem vários algoritmos desenvolvidos mas que por uma questão de relevância vão ser alvo de foco apenas dois: *Pre-Copy* e *Post-Copy*.

Para efeitos de comparação segue em baixo a Figura 2.20 que tem a forma "normal"ou clássica de fazer uma migração entre *VMs*.

[22]Um *thin client* é o nome que é dado a um sistema que apenas fornece um interface gráfica num *web browser* para o seu consumidor final utilizar.

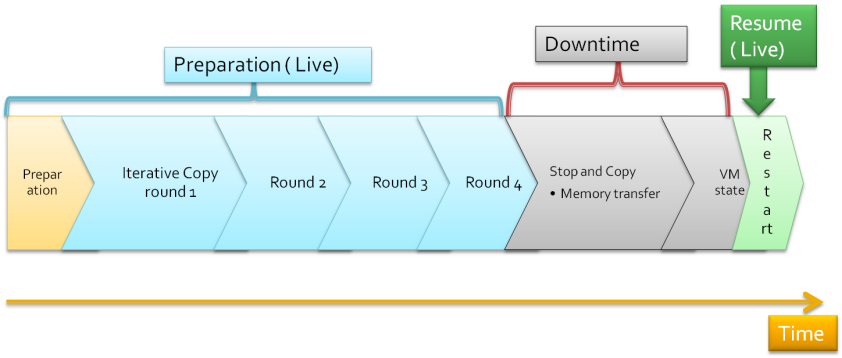

Figura 2.20: *Classic-Copy Algorithm*[4]

Ambos os algoritmos são baseados num conceito de migração de memória. Este conceito implica três fases: *push, stop-and-copy* e *pull*.

A primeira fase, *push*, é referente à passagem das páginas a ser utilizadas na *VM1* para a rede da máquina de destino[4]

A segunda fase, *stop-and-copy*, refere-se à paragem da máquina virtual de origem e à passagem dos restantes elementos para a máquina virtual de destino, consequentemente dá-se a inicialização da máquina "cópia"[4].

A terceira e última fase, *pull*, não é mais do que o *boot* da *VM2* e do *kill* da *VM1* [4].

2.3.2.1 *Pre-Copy*

Este algoritmo usa uma combinação de *push* e *stop-and-copy*[4] mas é constituído por um total de cinco etapas de execução que estão descritas na Figura 2.21.

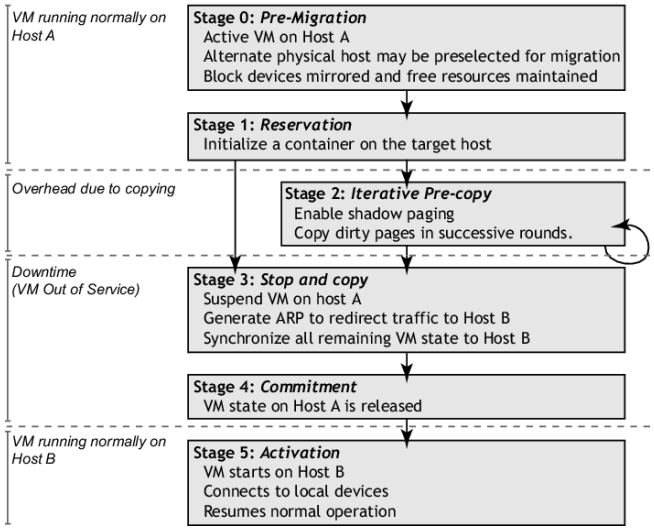

Figura 2.21: *Pre-Copy Algorithm*[4][5]

2.3.2.2 *Post-Copy*

Este algoritmo por "oposição"ao anterior utiliza uma combinação entre *pull* e *stop-and-copy*. [4] As etapas de execução estão presentes na figura 2.22.

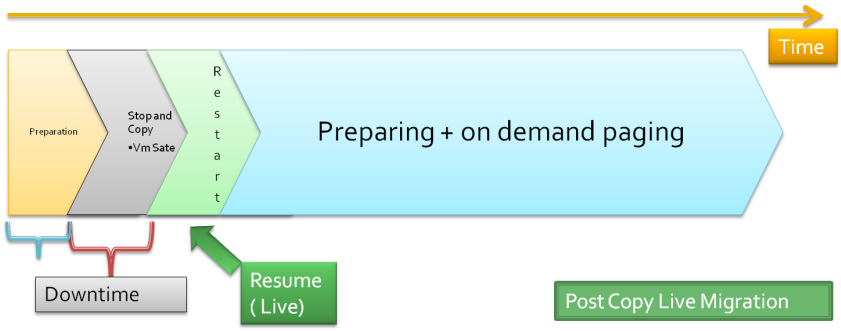

Figura 2.22: *Post-Copy Algorithm*[4]

2.4 IPBrick

IPBrick é uma solução empresarial que tem como principal vantagem e *slogan* a descomplexidade da instalação e configuração, a redução de custos, sendo que é o *software* com menos custos de propriedade do mercado. Esta é uma das razões pelas quais se está a espalhar rapidamente pelo mundo empresarial [45, p.25–216][46].

A *IPBrick* é um mundo de *appliances*, desde uma plataforma de comunicações até aplicações para empresas passando também pelo *hardware*. Conseguem-se encontrar possibilidades como: *Call Center, Digital Signage, HotSpot, Radio, Billing, Open Desktop, UCoip Recording*, entre muitas outras[47].

Todos estas aplicações listadas anteriormente estão dispersadas pelos módulos seguintes que serão, de seguida, aprofundados [45, p. 217–304].

Figura 2.23: *IPBrick*.

2.4.1 Produtos

As possibilidades do *software IPBrick* são variadíssimas por isso é necessário descreve-las um pouco, não só para se perceber o contexto em que a empresa se encontra e o que pode ser melhorado, mas também para se perceber o que é que uma atualização efetivamente atualiza assim que seja introduzida no sistema.

2.4.1.1 *.I*

O módulo I refere-se à *intranet*[23]. Este módulo é responsável pela configuração rápida e fácil da rede interna de um cliente e isto implica o sistema de *e-mail*, controlo do servidor de domínios, servidor de ficheiros e de áreas de trabalho individuais, servidor de impressões e mais[48][45, p. 25–304].

Fisicamente os *appliances* disponíveis são três:

1. I 16 - 50 *users*
2. I 32 - 100 *users*
3. I 64 - 200 *users*

2.4.1.2 *.GT*

O módulo *GT* é uma das razões pelas quais o *software IPBrick* é considerado tão revolucionário e tão distinto. Esta solução é baseada em *UCoIP*, que é o acrónimo para *Unified Communications over IP* que se traduz para Comunicações Unificadas sobre *IP*.

Que por sua vez traduz-se também em Chamadas, Video-Chamadas, *Fax*, *E-Mail*, *SMS* e outros [49], que são cruciais em qualquer tipo de empresas. Estarem todos aglomerados sobre o protocolo da *Internet* possibilita facilidade de configuração, baixo custo e mobilidade no mundo da telecomunicação[49][41].

Principais vantagens ao ter o *.GT* são:

- Mobilidade - Usar o escritório onde quiser quando quiser, não há preocupações com *roaming* desde que se possua uma ligação à *internet* graças à *VPN SSL feature*.
- Disponibilidade - Para além de se rever este ponto na Mobilidade tendo em conta que o serviço está sempre disponível vê-se também no atendedor interactivo e automático de chamadas.
- *Mail2SMS* - Permite enviar mensagens de texto através do *E-Mail* para qualquer número de telemóvel.
- *Messenger* - Servidor para *chat* de mensagens instantâneas.

Em qualquer outro bloco *IPBrick* tudo é personalizável e este não seria a excepção como tal seja o *Fax* e os seus parâmetros, os *users* a quem é permitido utilizar o serviço e servidor de *Fax* bem como todas as características das outras *features* disponíveis [45, p. 25–304][41].

Pertinente será também perceber como tudo isto é feito. Como o próprio nome subentende, de certa forma e para quem já estiver familiarizado com *VoIP*, os serviços *UCoIP* utilizam exatamente as mesmas redes para fazer a sua distribuição. Todas estas características e alterações estão à

[23] Uma *intranet* é uma rede local ou restrita de comunicações, geralmente associado a empresas que pretendem manter todas as suas comunicações completamente privadas para proteção de dados ou investigações.

distância de *clicks* na *interface web* que foi criada de forma a ser o mais intuitiva e *user friendly* possível [45, p. 217–304][49].

2.4.1.3 *.4CC*

O módulo da virtualização, *For Cloud Computing* ou simplesmente *4CC*, é o que permite os clientes da *IPBrick S.A.* desenvolver uma plataforma onde é possível instalar, gerir e criar múltiplos servidores virtuais através de uma *interface Web* simplista [50].

Vantagens do *4CC*:

1. Disponibilidade - O sistema está sempre disponível, mesmo que o servidor físico falhe. Havendo outro servidor na rede as máquinas virtuais podem ser migradas para esse mesmo e continuar em funcionamento [45, p. 217–304].

2. Redução de Custos e Conhecimento - Com este módulo evita-se a necessidade de manutenção custosa e constante dos servidores físicos bem como a necessidade de ter um administrador com grandes conhecimentos técnicos [45, p. 217–304].

3. Melhoramentos - Se o cliente for uma empresa que fornece *software* como um serviço o módulo *IPBrick.4CC* funciona como um *booster* uma vez que a empresa teria simplesmente os recursos disponíveis para o desenvolvimento sem nenhuma outra preocupação [50].

O *Hypervisor* que está disponível tem de ser activado para poder serem utilizadas as capacidades de virtualização da *IPBrick*, sendo que este *VMM* não é mais do que a *admin account* do *IPBrick.VDI* [45, p. 217–304] cuja informação está imediatamente abaixo.

O primeiro passo na configuração deste tipo de funcionalidades seria ativar as permissões do tipo *ACL* uma vez que isto permite mexer com as permissões das sub-pastas independentemente das permissões do *parent*. O segundo passo seria dar permissões a um pré-configurado Grupo de Acesso através do módulo *I*. Posto isto a possibilidade de partilhar *ISOs* ficaria disponível [51][45, p. 217–304].

2.4.1.4 *.VDI*

Esta secção do *software IPBrick* não é mais do que um sub-módulo contido no *4CC* onde o servidor *VDI* fornece um *OS* aos *thin clients* sendo apenas necessário efetuar configurações de:

1. Terminal;

2. *Boot* e *OS*;

3. Máquinas[24];

4. Cliente;

[24] As máquinas que estão registadas no módulo *I* após ser feito o iniciar do sistema operativo do controlador o cliente tem de escolher o *kernel* e *OS* que quer usar.

5. *Broker*[25];

2.4.1.5 .C

Ou *IPBrick.Communications* pode funcionar ao nível das comunicações como *VPN, Web, E-Mail Relay, Webmail, Intrusion Detection System, Proxy, etc*[41].

Em todos estes como já visto nos módulos anteriores existem várias opções de personalização associadas, tomando o *Proxy* como exemplo, este pode ser de três tipos:

- Normal - Caso pretendam configurar no *browser* a porta 3128 é a disponível para o *Proxy server* da *IPBrick* sendo que existe um registo dos acessos para propósitos estatísticos;
- Transparente - Grande parte da *Internet* funciona pela utilização de *Proxy* transparentes que agilizam a receção de conteúdos e as comunicações entre cliente-servidor, o utilizador continua a ter acesso à *Internet* mas a *firewall* redireciona-o para o *Proxy*;
- Autenticação - Acesso à *Internet* será apenas possível através da passagem e autenticação neste *Proxy* sempre que um *browser* abre é pedido um par *login/password*.

2.4.1.6 .SEC

O *SEC* não passa de um sub-módulo da secção *C* cujo objectivo é fornecer características de segurança a todo o sistema *IPBrick*. Previne, por exemplo, o acesso direto à *Internet* através das *workstations* bloqueando *Trojans* de estabelecer acessos remotos à *intranet* [45, p. 217–304].

Este sub-módulo não se resume à prevenção de acessos, tem outras soluções de segurança e as disponíveis são:

- Anti-Virus;
- Segurança da *Intranet*;
- *Anti-SPAM*[26];
- Prevenção de DoS[27];
- *Firewall*
- Servidor *VPN*[28].

[25] O *Broker* é uma ferramenta incorporada neste módulo que aquando da activação permite fazer o balanço automático tendo em conta várias possibilidades: memória, *CPU*, servidores[45, p. 217–304].

[26] Refere-se, de forma muito generalista, a não solicitadas ou indesejadas mensagens de correio eletrónico.

[27] *Denial-of-Service* é um ataque que tem como objetivo provocar a indisponibilidade de um sistema ou parte de um sistema que está a ser utilizado ou viria a ser utilizado por um grupo especifico de pessoas.

[28] *Virtual Private Network* rede privada geralmente associada para acesso remoto a uma rede.

2.4.1.7 *.CAFE*

Uma empresarial rede social tem como objetivo fomentar a troca de informação e as relações entre os colaboradores, o que consequentemente aumentará a produtividade.

O *IPBrick.CAFE* foi desenvolvido de forma a que o parágrafo anterior tivesse evoluído para um outro nível aplicando o sistema *UCoIP* integrado na rede social bem como acesso direto às aplicações de negócio.

O *CAFE* permite:

- Comunicação direta de 4 formas: *email*, *chat*, chamada e video.
- Partilha de documentos[29].
- Video-chamada através de *Web Browser*.
- *Feed* de Notícias.
- Área Empresarial e Pessoal.
- Notificações e Perfis.

2.4.1.8 *iPortalDoc*

iPortalDoc é a solução utilizada para gerir documentos e processos baseando-se nos fluxos de trabalho. Esta ferramenta permite integrar a modelação e implementação dos processos públicos ou privados, com acesso a todo o histórico para as pessoas implicadas no processo, com os documentos, *emails* e com o sistema *UCoIP* disponível.

No *iPortalDoc* é possível conjugar:

- Faturação Eletrónica com Assinatura Digital;
- Encomendas e Compras;
- Propostas;
- Correspondência;
- Contratos e Reclamações.

Com todo o sistema de comunicações que fica gravado no *iPortalDoc* onde tudo fica associado ao processo:

- Chamadas;
- Emails;
- Conversas de *Chat*;

[29] Os documentos podem ser partilhados a partir do *CAFE* clicando apenas num botão.

2.4.2 Características *IPBrick OS*

Nesta secção são apresentadas algumas características do sistema operativo, nomeadamente o esquema de partições e a montagem das partições nos diretórios, a interface gráfica e principais diferenças entre a *IPBrick v5.3* e a *IPBrick v6.2*, a última versão que foi disponibilizada.

A *IPBrick* dá aos seus clientes a garantia de que os dados que estes possuem dentro do *OS* não são alterados nas operações que envolvam a modificação do respetivo sistema portanto é extremamente necessário e pertinente na atualização de uma distribuição para outra, e considerando que a nova distribuição tem um tipo de ficheiro diferente, é necessário migrar ou então manter no caso de ser compatível. O mesmo se aplica ao esquema de partições, Figura 2.25 e 2.26.

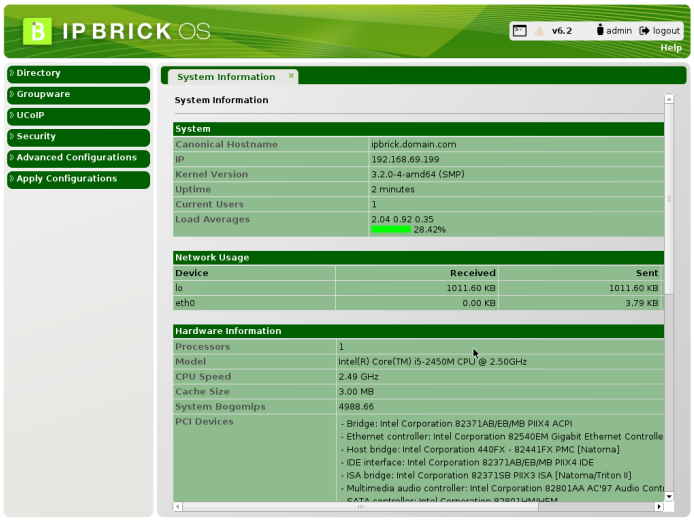

Figura 2.24: Informação de Sistema - *IPBrick v6.2*.

2.4.2.1 Partições

Neste momento o esquema de partições é baseado na separação das componentes importantes do sistema, ou seja, */, usr, var, opt, home1, home2, swap* sendo que tanto os diretórios *usr, var, home1* e *home2* pertencem, na versão 5.3 da *IPBrick*, a uma partição dita secundária, Figura 2.25, partição essa que é como se fosse virtualizada resultando nas quatro partições descritas.

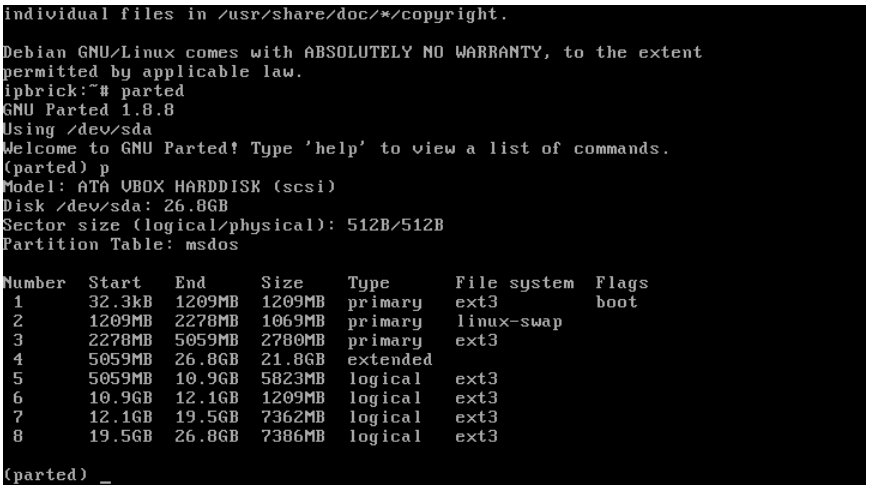

Figura 2.25: *Partition List IPBrick v5.3*.

As versões a partir da 6.0 não incluem esta partição secundária passando a ser todas primárias, tal como se pode ver na Figura 2.26, o que permite uma maior versatilidade ao sistema no que toca a apagar, criar, redimensionar e editar partições.

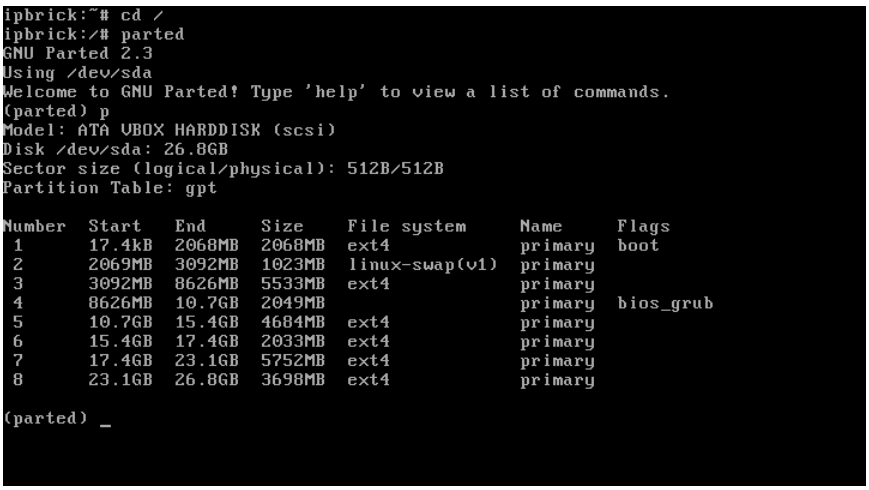

Figura 2.26: *Partition List IPBrick v6.2*.

Como apresentado na secção 2.1.4 as nomenclaturas associadas ao sistema vão mudando com as diversas variáveis (disco, controlador) portanto a Figura 2.27 embora mostre uma certa nomenclatura de como o sistema está no momento montado não é necessariamente assim que se apresentam todos os sistemas.

```
ipbrick:/# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,relatime,size=10240k,nr_inodes=255208,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmod
e=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=206128k,mode=755)
/dev/disk/by-uuid/a8956c03-38da-4cae-b191-217ec16e56ab on / type ext4 (rw,relati
me,errors=remount-ro,user_xattr,barrier=1,data=ordered)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /run/shm type tmpfs (rw,nosuid,nodev,noexec,relatime,size=612120k)
/dev/sda3 on /usr type ext4 (rw,relatime,user_xattr,barrier=1,data=ordered)
/dev/sda5 on /var type ext4 (rw,relatime,user_xattr,barrier=1,data=ordered)
/dev/sda6 on /opt type ext4 (rw,relatime,user_xattr,barrier=1,data=ordered)
/dev/sda7 on /home1 type ext4 (rw,relatime,user_xattr,barrier=1,data=ordered,usr
quota,grpquota)
/dev/sda8 on /home2 type ext4 (rw,relatime,user_xattr,barrier=1,data=ordered,usr
quota,grpquota)
/dev/sda7 on /var/spool/hylafax type ext4 (rw,relatime,user_xattr,barrier=1,data
=ordered,usrquota,grpquota)
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
ldap://127.0.0.1/ou=auto.home,dc=domain,dc=com on /home type autofs (rw,relatime
,fd=5,pgrp=3330,timeout=300,minproto=5,maxproto=5,indirect)
nfsd on /proc/fs/nfsd type nfsd (rw,relatime)
ipbrick:/# _
```

Figura 2.27: *Mount List IPBrick v6.2*.

2.4.2.2 Interface Gráfica

A interface gráfica da *IPBrick* foi sofrendo variadíssimas alterações ao longo do tempo, Figuras 2.28, 2.29 e Figura 1 do Anexo 5.2, embora a base se mantenha constante e extremamente intuitiva.

Esta interface consiste num menu do lado esquerdo onde é possível chegar às definições que se pretende visualizar ou editar. Onde após se abrir uma das opções aparece em cima, sob a forma de *tab*, as opções escolhidas, a informação disponível e o que é editável ou não através de botões como *Insert* e *Modify*.

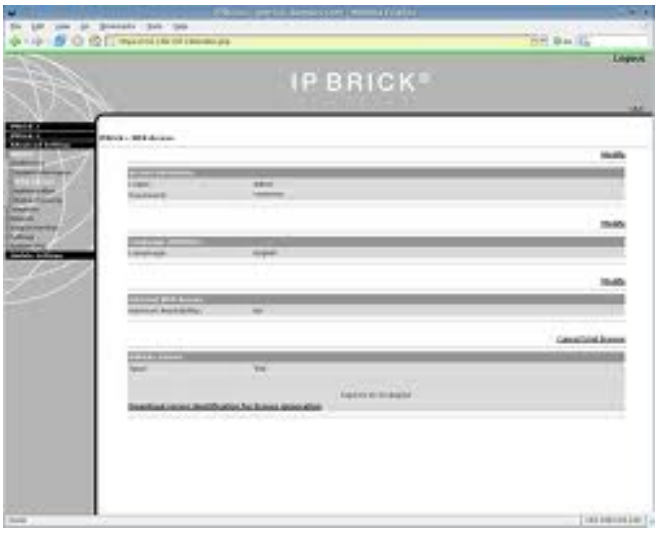

Figura 2.28: *GUI IPBrick v5.3*.

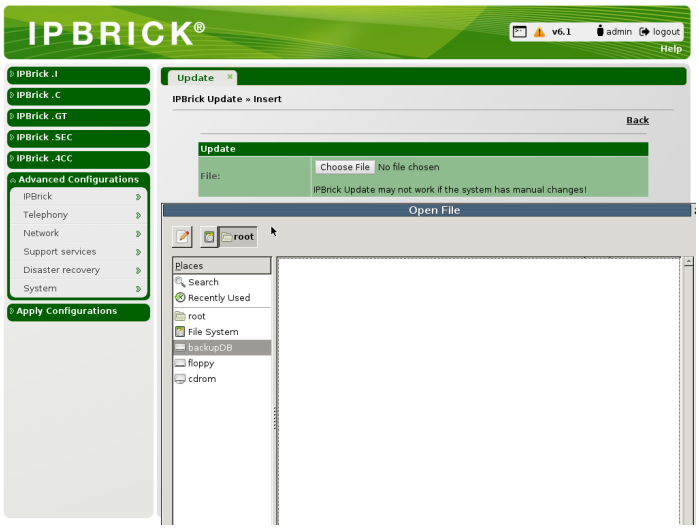

Figura 2.29: *GUI IPBrick v6.1*.

2.4.2.3 Processo de Instalação

Para se realizar a instalação do *software* num sistema físico neste momento são necessárias as seguintes características:

- *DVD, USB* de instalação;
- Mínimo de 2 *GB* de *RAM*;
- Mínimo de 20 *GB* de *HDD*;
- *NIC* - Placa de Rede.

As etapas do processo de instalação embora simples, acabam por ser fatigantes para qualquer cliente que pretende apenas utilizar o *software* diretamente e nada mais. É exatamente aqui que se cifra o objeto deste trabalho: anular a no tempo que demora a fazer uma instalação tornando-a o mais automatizada possível nas atualizações.

1. *Boot* na *BIOS* a partir do *DVD*[52, p. 25–34].
2. *Install Option* - Opção para uma instalação inicial, verifica também se já existem versões anteriores do sistema. Se sim, gera *prompt* para existir um *backup* dos dados gerados até então.
3. *ReInstall / Upgrade* - Estando já na versão 6.0 pode haver nas instalações anteriores diferentes tipos de ficheiros que não o *EXT4* portanto o que o processo faz é manter as partições de dados no formato anterior e as partições de sistema no novo formato[52, p. 25–34].
4. *Advanced Install* - Tem várias opções que podem ser verificadas em [52, p. 9–24].

5. *Create Bootable Pendrive* - Cria uma versão portátil para uma *USB Pen* através da criação de uma imagem a partir do *DVD*.

6. Manual - Como o próprio nome indica, trata-se da instalação manual, nomeadamente através da linha de comando.

Após se dar início à instalação propriamente dita é apresentado uma barra de progresso que seria a apresentação *default* ou então texto. Caso seja encontrada outra versão do sistema ou outro sistema operativo é gerado um alerta onde o utilizador deverá escolher se pretende continuar com a instalação, fazendo um *update* ou então instalando o *IPBrick* paralelamente [52, p. 9–24][52, p. 25–34].

2.5 Atualizações de *software*

As atualizações de *software* estão dividas entre *major* e *minor* independentemente de existir uma interface gráfica e um nível de automatismo associado.

Major existe quando é necessário fazer uma mudança "drástica"ou reescrever muitas partes de um sistema, exemplos disto são: a passagem do *Windows 7* para o *Windows 8* ou a passagem do *OS X Lion* para o *OS X Mavericks* ou a passagem do *Debian Lenny* para o *Debian Wheezy*[53].

Minor existe quando é necessário de adicionar alguma parte do sistema ou da aplicação e que o objetivo é melhor performance, estabilidade, segurança, etc. Trata-se de uma modificação pequena do objeto que está a ser tratado, como por exemplo: *Windows 8* para *Windows 8.1* ou *Google Chrome 33.0.0.1* para *Google Chrome 33.0.0.2*[53].

2.5.1 Empresas

Tanto a *Amazon* como a *Microsoft* são empresas que oferecem soluções empresariais para outras empresas sobre máquinas virtuais *Linux* e ambas possuem *software* que permite a instalação de *updates* nos sistemas operativos emulados através das *VM*. Nesse sentido encontram-se referenciadas de seguida para que se possa obter uma base de comparação.

2.5.1.1 *Amazon*

Nos serviços da *Amazon* relativos ao *Linux*, sendo este serviço de nível mais baixo, geralmente *IaaS*, "obrigam"o utilizador a instalar *packages* manualmente através do terminal por ligação primeiro a um repositório central denominado de *Amazon Linux* e caso não estejam todos os *packages* disponíveis nesse, aconselham ao cliente a utilizar outros repositórios também da *Amazon* ou instalação direta na sua instância.

No caso de *updates* de segurança, *bugfix* (que incluem os anteriores) ou todo o tipo de atualizações funciona na inicialização da máquina. Apesar de ser feito ao nível do terminal existe um certo automatismo e controlo, uma vez que o *download* e instalação destas atualizações é feita quando a

máquina é lançada. Por predefinição os *updates* que são feitos são os de segurança mas é permitido ao utilizador modificar estas definições através do comando *cloud-config*[54, p. 119–121].

2.5.1.2 *Microsoft - Azure*

O *Azure* faz os seus *updates* de uma forma diferente do *Windows* normal, em vez de ir lançando pequenos *updates* que vão sendo instalados automaticamente através de um módulo que permite o *download* e instalação por agendamento ou automaticamente, gera novas imagens contendo as últimas atualizações e simplesmente faz *boot* com as novas atualizações inseridas. O *Azure* possui também um mecanismo que permite salvaguardar que os serviços do cliente não ficam em baixo totalmente, ou seja, supondo que existe mais do que um servidor que desempenha a mesma função estes devem ser agregados num *Availability Set* de forma a que o programa responsável pelos *updates* possa aperceber-se deste caso particular e faço os servidores entrar em manutenção em diferentes momentos, desta forma permite-se que o cliente mantenha sempre algum tipo de funcionamento [55, 56].

A grande desvantagem de utilização do *Azure* é o facto deste não permitir o controlo por parte do utilizador dos *updates*. O utilizador não pode escolher evitar um *update* ou o seu serviço não ser interrompido, no entanto, garantem que caso o utilizador tenha pelo menos duas instâncias a disponibilidade das *VMs* é de 99,95%. Impõe também ao utilizador que ocorram dois *reboot* todos os meses devido a *updates* em primeiro lugar no *Host* e depois no *Guest*, onde o utilizador incide. No entanto este segundo *reboot* (devido ao *Guest OS*) só acontece caso o utilizador tenha os *updates* automáticos ligados[57, 56].

2.5.2 *Linux*

No *Linux* as atualizações podem ser feitas utilizando várias ferramentas, que podem ser gráficas ou *CLI*[30]. No caso das que são baseadas em *bash*, Secção 2.1.2.3, existem: *dpkg*, *apt* ou *aptitude*. Em termos de ferramentas gráficas existem: *GNOME Software* e *Synaptic*, que fazem parte das *GUI GNOME* e *KDE*, respetivamente. Estas atualizações podem assumir três formas, independentemente do tipo de *deployment* utilizado: *distribution updates* ou *major*, *version updates* ou *minor* e ainda *updates* das applicações instaladas.

Embora se faça uma distinção vincada entre *PMS*, *Package Management Systems* gráficos e com recurso à linha de comandos é importante perceber que os gráficos apresentam uma interface para que para o utilizador comum que não possui conhecimentos avançados sobre o sistema ou que pretende ter uma solução mais cómoda consiga sempre obter o funcionamento mais *user-frendly* e eficiente possível mas na verdade os *PMS* gráficos não são mais do que um encapsulamento de alto nível dos *CLI*. Por outras palavras, quando se desce de nível as funções executadas são as do *apt, dkpg, aptitude*, comandos como os que estão expostos na Tabela 1 do Anexo 5.2. As Figuras 2.30 e 2.31 mostram um pouco da interface gráfica.

[30] *Command Line Interface*, traduzindo linha de comandos.

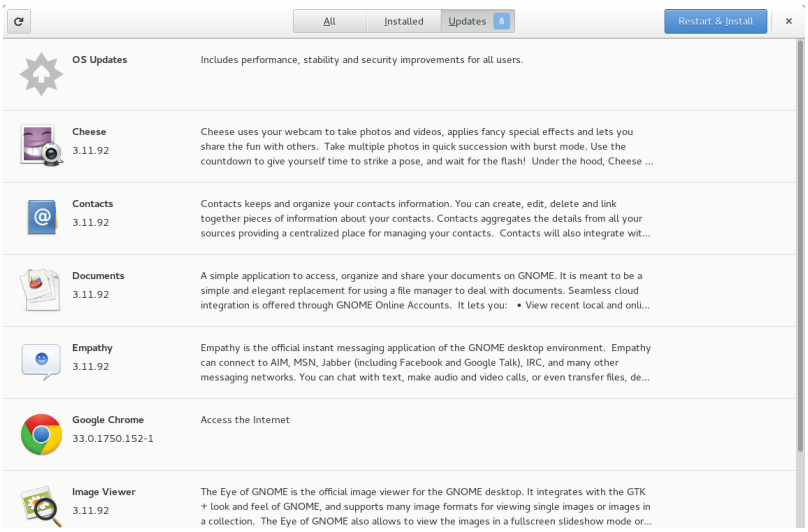

Figura 2.30: *GNOME software*.

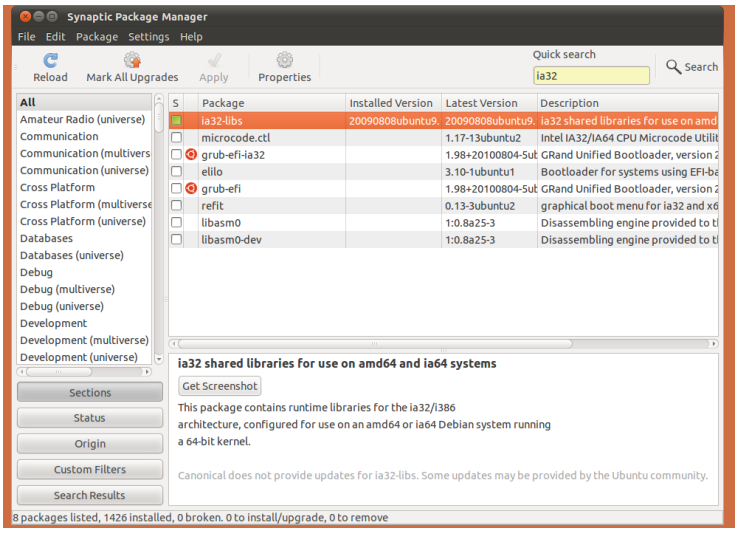

Figura 2.31: *KDE Package Management System GUI*.

2.5.3 *IPBrick*

2.5.3.1 *Minor Updates*

Todos os *Updates* gerados para a versão 6.1, a última versão disponível do *software IPBrick*, e mesmo outro tipo de *updates* são feitos manualmente, ou seja, o cliente tem de primeiro fazer o *download* da nova possibilidade, de seguida, tem de aceder à Interface Gráfica *Web*, e introduzir o ficheiro *.deb* que descarregou sob o painel: *Advanced Configurations -> IPBrick -> Update -> Insert*. O utilizador deverá fazer isto, por ordem, para todas as atualizações, *minor updates*,

mesmo que estas sejam simples *bug fixes* e *security updates*. Após a introdução de todas é preciso ainda aplicar as novas configurações, *Apply Configurations*.

O processo é válido para todas as versões existentes do *software* e impõe ao utilizador que anteriormente a este poder passar de uma versão 5.3 para 5.4, por exemplo, seja necessário introduzir todos os *updates* devido à recursividade existente no *Linux*.

A Figura 2.32 permite ver como se instala atualmente um *minor update* no sistema operativo *IPBrick*. Este método é transversal a todas as versões.

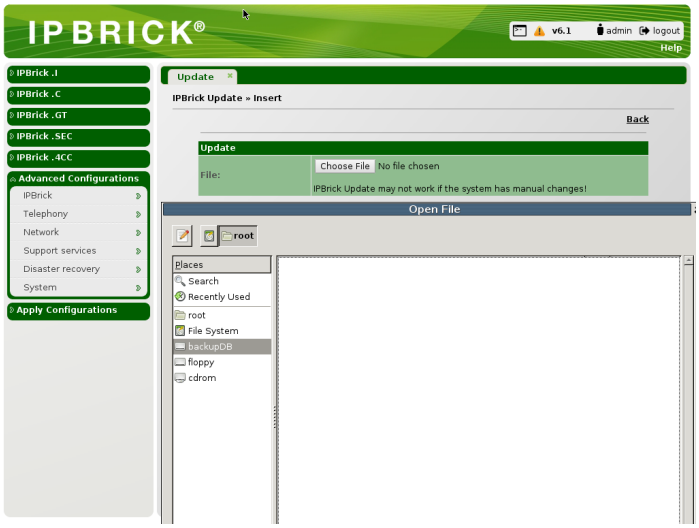

Figura 2.32: Painel de inserção de pacotes de atualização *IPBrick-v6.1*.

2.5.3.2 *Major Updates*

As atualizações para uma distribuição diferente no *software IPBrick* são apenas possíveis, neste momento, através de um novo processo de instalação através de um *CD* ou *DVD* ou de uma pen *USB*. Este processo implica reescrever todo o disco tal como numa nova instalação.

Como uma das imagens de marca da Empresa trata-se do facto de os utilizadores manterem sempre os dados sem que estes sofram qualquer tipo de interferência, as imagens *ISO* geradas incluem um algoritmo de reconhecimento da existência de uma versão *IPBrick* anterior. No caso de se confirmar a existência, a rotina de instalação automaticamente passa as partições, correspondentes às pastas (*homeY*) e portanto aos dados do utilizador, à frente não as reescrevendo garantido que a imagem de marca se mantém.

Isto funciona do ponto de vista *indoor*, onde os servidores estão dentro de portas e os clientes conseguem controlar diretamente o sistema, no entanto, em clientes que utilizam servidores e máquinas de *Cloud* o mesmo não acontece uma vez que não existe *hardware* onde introduzir os meios físicos de instalação.

2.6 Linguagens

Na criação ou alteração de interfaces gráficas ou Bases de Dados e outras ferramentas necessárias ao funcionamento de uma interface *web* são necessários conhecimentos em diversas linguagens de programação sendo portanto pertinente referir as respetivas ferramentas neste documento, apresentam-se de seguida alguns conceitos sobre cada uma das linguagens e para que cada uma é utilizada.

2.6.1 Interface Gráfica *Web*

A interface gráfica *web* pode ser desenvolvida de várias formas e com recurso a várias linguagens tais como: *HTML, CSS, JavaScript, PHP, Bootstrap* e embora se referencie todas estas opções apenas os três primeiros são consideradas os pilares, ou a Tríade, de *Web Design* e por isso considerados os mais importantes.

2.6.1.1 *HTML*

HTML é o acrónimo de *Hyper Text Markup Language* e é a linguagem responsável pelo conteúdo existente nas páginas *web* na *World Wide Web*, ou *www*. Esta linguagem consiste na utilização de *markups/tags* ou etiquetas, que modificam o estilo do conteúdo existente na página, e na integração de *links*, que permitem ligar as diversas páginas entre si [58, 59].

O *HTML* é a ferramenta que permite definir tamanhos de letra, tipo de letra, cores, parágrafos, títulos e muitas outras características visuais. Para uma noção visual de um código exemplo desta ferramenta é necessário ver a Figura 2.33 ou a Figura 2.37, onde se encontra *PHP* também e a secção 2.6.1.4 que o explica.

```
1
2  <!DOCTYPE html>
3  <html>
4  <body>
5
6  <h1>My First Heading</h1>
7
8  <p>My first paragraph.</p>
9
10 <a href = "www.franciscooliveira
11     thesis.ipbrick.com"> Thesis </a>
12
13 </body>
14 </html>
15
```

Figura 2.33: *HTML* - código exemplo.

2.6.1.2 *CSS*

Cascading Style Sheets é o significado do acrónimo *CSS* e esta linguagem de programação tem como principal objetivo simplificar a aplicação do estilo de uma página e as alterações necessárias *a posteriori*. Através do *CSS* é possível estabelecer prioridade, hereditariedade ou tornar o *design* uma cascata. Isto quer dizer que são listadas regras numa página global e que todas as páginas que derivarem dessa cumprem essas mesmas diretrizes o que torna tudo muito mais automatizado e sem a necessidade de alterar tudo de um documento em cada documento. A Figura 2.35 permite perceber através do código exemplo e das *tags:* <*style*></*style*> como o *CSS* é representado e a Figura 2.34 o resultado da utilização[60, 61].

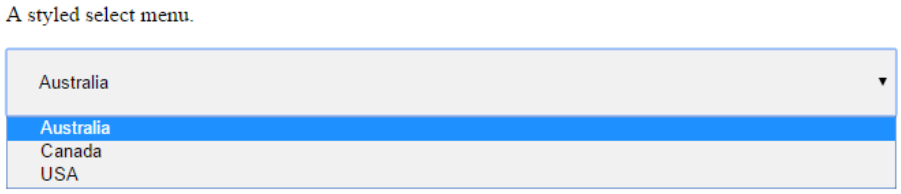

Figura 2.34: *CSS* - Menu selecionável.

Sendo escalável permite que haja uma separação vincada entre *HTML*, passa a ser específico a conteúdo, e *CSS*, específico para estilo, o que promove o acesso à informação, a redução da complexidade, o controlo sobre a apresentação e a adaptação aos diferentes tipos de plataformas e apresentações, seja no ecrã, através de impressão, seja num *tablet* ou num *smartphone*[60, 62, 63].

```
<!DOCTYPE html>
<html>
<head>
<style>
select {
    width: 100%;
    padding: 16px 20px;
    border: none;
    border-radius: 4px;
    background-color: #f1f1f1;
}
</style>
</head>
<body>

<p>A styled select menu.</p>

<form>
  <select id="country" name="country">
    <option value="usa">Australia</option>
    <option value="usa">Canada</option>
    <option value="usa">USA</option>
  </select>
</form>

</body>
</html>
```

Figura 2.35: *CSS* - Código Exemplo.

2.6.1.3 *JavaScript*

JavaScript é a linguagem associada à imposição de regras e à gestão do comportamento das páginas *Web*, por outras palavras, é através desta ferramenta que, por exemplo, é possível fazer uma publicação numa rede social sem a necessidade de refrescar a página ou transmitir informação sobre os hábitos de leitura do utilizador[64, 65]. Isto só é concebível devido às seguintes características:

- Dinâmica[31].
- Orientada ao Objeto.
- Não lida com *I/O*[32].
- Sintaxe derivada de *Java*[33].

Embora seja principalmente característica do lado do cliente e no contexto *Web* não são apenas estas as aplicações do *JS*, este também se encontra em:

- *PDF - Portable Document Format*
- *Desktop Widgets*
- Máquinas Virtuais
- Desenvolvimento de Jogos
- Aplicações para *Mobile* e *Desktop*

Na Figura 2.36 pode observar-se algum código exemplo simples da sintaxe do *JavaScript*:

```
1
2  var sum = function() {
3      var i, x = 0;
4      for (i = 0; i < arguments.length; ++i) {
5          x += arguments[i];
6      }
7      return x;
8  }
9  sum(1, 2, 3); // returns 6
```

Figura 2.36: Somatório - código exemplo.

[31] Devido a este dinamismo é possível criar o *Dynamic HTML*.
[32] Não tem mecanismos para tratamento de *Input / Output*, esse fica a cargo do ambiente no qual o *JS* está inserido.
[33] *Java* é outra linguagem de programação de médio-alto nível, utilizada em grande parte no desenvolvimento de aplicações, jogos e outros programas, pouca relação tem com *JavaScript* exceto no nome.

2.6.1.4 *PHP*

PHP é um acrónimo recursivo de *PHP: Hipertext Preprocessor* e é uma linguagem de programação utilizada no desenvolvimento de páginas *web* em conjunto com *HTML*. Com recurso ao *PHP* é possível gerar *HTML* do lado do servidor que posteriormente será enviado para o cliente sem que este saiba exatamente que código foi executado, cria *scripts* que evitam que se tenham de utilizar variadas *tags* de *HTML*[66].

O código presente na Figura 2.37 representa uma rotina que imprime "Olá Mestrando, acabaste de usar PHP!"e outra que imprime os números de um até 10 (exclusive), Figura 2.38.

```
<!DOCTYPE HTML>
<html>
<head>
    <title>Código Exemplo</title>
</head>
<body>
    <?php
        echo "<p> Olá Mestrando, acabaste de usar PHP! </p>";
    ?>
    <h1> Isto é um paragrafo HTML </h1>
    <?php
       for($i=1; $i<10; $i++){
            echo "<p>". $i ."</p>";
        }
    ?>
</body>
</html>
```

Figura 2.37: *PHP* com *HTML* - código exemplo.

Olá Mestrando, acabaste de usar PHP!

Isto é um paragrafo HTML

1 2 3 4 5 6 7 8 9

Figura 2.38: *PHP* - resultado exemplo.

2.6.2 Bases de Dados

Uma Base de Dados, ou *DB*, é definida como uma coleção de informação que está categorizada e organizada de forma a que possa ser gerida, trabalhada, acedida e atualizada de forma mais prática, rápida e simples. Podem classificar-se pelo tipo de dados que tratam ou pelos modelos lógicos de organização e gestão da informação, como por exemplo:

- Hierárquico

- Entidade-Associação
- Orientado a Objetos
- Relacional
- *XML*[34]

Embora existam todos estes modelos possíveis o mais usado, quase de forma unânime, trata-se do modelo relacional e por isso é neste que se exerce o foco desta secção.

O modelo relacional baseia-se nos ramos da matemática: teoria dos conjuntos e lógica. Uma *DB* trata-se de uma quantidade aleatória de dados onde são aplicadas operações, ou filtros, que retornam tabelas ordenadas mediante as relações impostas pelos filtros. O significado de relacional neste caso é a identificação de padrões entre conjuntos de informação, de uma forma mais concreta diz-se que um objeto ou conjunto A está relacionado com um objeto ou conjunto B se após comparação estes exibem padrões comuns [67].

2.6.2.1 *Structured Query Language (SQL)*

SQL como a própria tradução do acrónimo significa, Linguagem de Consulta Estruturada, constitui uma série de normas que quando aplicadas uma *DB* relacional através de um *DBMS* (ou *RDBMS*) resultam num conjunto de informações que tem as características utilizadas para a pesquisa em comum [68, p. 7–15].

No que esta linguagem difere de todas as outras, como *C, Java, Assembly*, é na maneira como processa os dados. Contrariamente às outras linguagens, que processam a informação unidade a unidade, o *SQL* processa o mesmo volume de informação por grupos o que o torna muito mais eficiente para grandes volumes de dados [69].

Esta forma de tratar os dados permite que o resultado de um filtro seja passado diretamente a outro comando, a uma aplicação ou ao utilizador diretamente num único passo.

Outras vantagens da linguagem:

1. Inserir, Atualizar ou Apagar blocos de dados
2. Criar, substituir ou alterar objetos
3. Controlar acessos
4. Garantia de consistência e integridade
5. Comum a todas as bases de dados relacionais

[34] *eXtensible Markup Language*, trata-se de uma linguagem que define regras para a encriptação, armazenamento e transferência de informação de uma forma legível para seres humanos e para máquinas.

2.6.2.2 *eXtensible Markup Language (XML)*

Esta linguagem permite declarar objetos que visam ser lidos tanto por seres humanos como por máquinas, ou seja, permite apresentar dados impondo um certo comportamento que a máquina tem de ter para apresentar os dados. Constitui uma forma de definir um conjunto de regras sobre como um *set* de dados deve ser tratado.

Documentos *XML* são constituídos por unidades de armazenamento onde são guardados dados de dois tipos: analisados ou *unparsed*. A informação analisada constitui texto ou etiquetas, *markups*, sendo que estes últimos impõe o *design* do armazenamento bem como a sua estrutura lógica de funcionamento. Através de da aplicação *XML processor* é possível depois aceder à informação e à estrutura do documento ou documentos *XML*.

Por analogia ao sistema de ficheiros e da rede de diretórios em forma de árvore do *Linux*, também o *XML* possui um diretório "raíz"de onde parte toda a estrutura e lógica do *fetch* da informação. Por exemplo, pode indicar-se o comportamento que um servidor tem de ter quando um cliente pretende fazer *download* de um ou mais ficheiros específicos, o cliente informa o servidor de algumas características suas e o servidor através do mapeamento feito e das caraterísticas recebidas sabe quais são os ficheiros que tem de fornecer e sobre que ordem, formato ou forma tem de o fazer, as Figuras 2.39 e 2.40 mostram um esquema de funcionamento do *XML* e um código exemplo respetivamente.

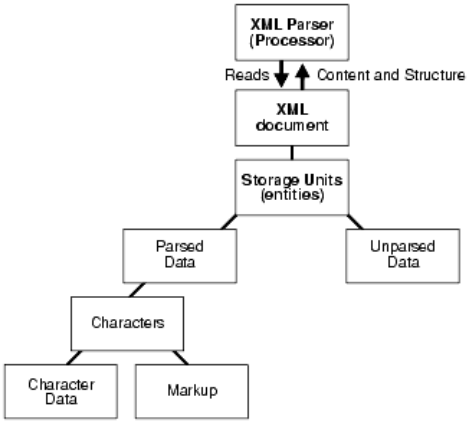

Figura 2.39: *XML* - Esquema exemplo.

```
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
      Two of our famous Belgian Waffles with plenty of real maple syrup
    </description>
    <calories>650</calories>
  </food>
  <food>
    <name>Strawberry Belgian Waffles</name>
    <price>$7.95</price>
    <description>
      Light Belgian waffles covered with strawberries and whipped cream
    </description>
    <calories>900</calories>
  </food>
  <food>
    <name>Berry-Berry Belgian Waffles</name>
    <price>$8.95</price>
    <description>
      Light Belgian waffles covered with an assortment of fresh berries and whipped cream
    </description>
    <calories>900</calories>
  </food>
  <food>
    <name>French Toast</name>
    <price>$4.50</price>
    <description>
      Thick slices made from our homemade sourdough bread
    </description>
    <calories>600</calories>
  </food>
  <food>
    <name>Homestyle Breakfast</name>
    <price>$6.95</price>
    <description>
      Two eggs, bacon or sausage, toast, and our ever-popular hash browns
    </description>
    <calories>950</calories>
  </food>
</breakfast menu>
```

Figura 2.40: *XML* - código exemplo.

2.6.2.3 *DataBase Management Systems (DBMS)*

DBMS é o *software* responsável por fazer a gestão de grandes volumes de dados quando o sistema de ficheiros não consegue ser capaz de o fazer de forma eficiente. Para que se perceba melhor o quão indispensável o *DBMS* é nestas situações pegue-se no exemplo de uma empresa qualquer que tem um certo número de funcionários [70, p. 5–9]. Cada um destes funcionários gera uma grande quantidade de dados todos os dias.

Em primeiro lugar, a quantidade de informação gerada não interessa para o *DBMS* uma vez que consegue sempre guardar a informação sem ter necessariamente de a dividir como num *filesystem* e não necessita de mapeamento uma vez que consegue encontrar qualquer dado necessário através de filtros.

Em segundo lugar, conseguindo encontrar os dados através de filtros não necessita de programas específicos, como nos sistemas de ficheiros, independentemente do volume de informação para as perguntas dos utilizadores, o utilizador pesquisa pela característica que pretende e os dados apresentados são todos os que têm relação com essa caraterística ou combinação de características [70].

Em último lugar, nem todos os utilizadores podem ter acesso a toda a informação contida na *DB* ou podem ver a informação mas não podem edita-la, esta diferença de privilégios pode ser implementada através de perfis de utilizador enquanto que num sistema de ficheiros tradicional apenas existe mecanismos de *password* para impedir acessos indesejados [70].

Concluindo, deste exemplo retira-se as seguintes vantagens da utilização de *DBMS*:

1. Independência dos dados
2. Acesso à informação Eficiente
3. Segurança e Integridade dos dados

4. Permite administração dos acesso feitos aos conjuntos e subconjuntos de informação
5. Agendamento de acessos concorrentes
6. Recuperação de Falhas
7. Fácil desenvolvimento de aplicações

PostgreSQL Trata-se de um *DBMS* orientado a objetos de forma relacional num sistema servidor/cliente que permite criar, apagar e gerir múltiplas bases de dados, perfis de utilizador e executar comandos *SQL*.

Isto quer dizer que existe a criação de um processo servidor e um processo cliente, onde o primeiro faz a gestão da base de dados e as operações em nome dos clientes após aceitar as suas ligações, e onde o segundo corre uma aplicação que pretende aceder a um conjunto de informações. Toda esta comunicação é feita através de uma ligação *TCP/IP* no caso de o cliente e o servidor serem duas máquinas diferentes [68, p. 1–5].

Apenas em novas ligações ao servidor é que o processo original, *postgres*, entra em funcionamento porque em todas as outras alturas fica apenas em segundo plano à espera de novas ligações e a as que estiverem em vigor são geridas pelos processos gerados a cada ligação.

Os perfis de utilizador são geridos separadamente dos utilizadores do sistema operativo, isto quer dizer que sempre que se tenta iniciar uma ligação a uma *DB* ela é feita com recurso ao *user* que for definido sendo que caso nenhum seja definido o *PostgreSQL* assume o utilizador do sistema como predefinido. Como resultado desta predefinição, a conta de sistema vai estar sempre estar registada como conta no servidor e esta tem sempre as permissões necessárias para criar mais *DB*.

2.7 Conclusão

Neste subcapítulo fica descrito os tipos de virtualização que podem ser encontrados numa *cloud*. O *software IPBrick* pode ser encontrado em vários tipos de mercados tecnológicos devido ao grande de número de *appliances* que fornece como tal foi necessário o desenvolvimento deste assunto para se poder introduzir o conceito de nuvem já de seguida onde assenta o *software* em questão.

Neste subcapítulo fica descrido não só as bases da *Cloud* onde assenta a *IPBrick* bem como técnicas de *Cloud Computing* que são utilizadas para minimizar o *downtime* dos serviços. É possível que no âmbito do desenvolvimento da tese seja necessário uma analogia ou até mesmo utilização destes mesmos algoritmos ou algum similares com o objetivo de desenvolver uma forma de minimizar ao máximo o tempo em que o serviço está indisponível durante uma atualização.

Todos estes mecanismos terão de ser explorados a fundo de forma a conseguir perceber-se como funcionam para verificar a possibilidade de usar como ponte ou base para o projeto em desenvolvimento.

Será necessário um conhecimento aprofundado de todas os módulos de forma a que na construção do módulo de gestão de atualizações se consiga estabelecer uma relação orgânica com todos

os outros módulos de forma a que as atualizações sejam o mais perfeitas possível. Ao conseguir-se a produção deste módulo, consegue-se também, como dito anteriormente, maior valor para a empresa e com certeza isso irá refletir-se no desempenho em termos de mercado.

Capítulo 3

Módulo para Gestão de Atualizações de *Software IPBrick*(MGA*SI*)

O Módulo para Gestão de Atualizações de *Software IPBrick*, doravante MGA*SI*, pode dividir-se em três grandes trabalhos tendo em conta os objetivos requeridos pela empresa. Interface Gráfica, atualizações de uma distribuição para a outra, ou *Major Updates*, e atualizações dentro da mesma distribuição, ou *Minor Updates*. Isto implica que sejam pensadas soluções para cada trabalho e são estas que são apresentadas de seguida bem como os resultados das mesmas.

3.1 Limitações e Algoritmia

3.1.1 Limites

Este Módulo de Gestão de Atualizações tem uma série de limitações, não só a nível do sistema sobre o qual o *software* foi construido mas também a nível empresarial:

- A passagem tem de ser válida tanto para utilizadores *indoor* como para utilizadores da *Cloud*;
- A atualização tem de ser *user-friendly*, ou, por outras palavras, tem de constituir algo muito simples e pouco intrusivo, ex.: carregar num botão de *update* ou correr um programa para atualizar através de uma pen ou através de download;
- Os dados do utilizador não podem ser comprometidos, toda a sua integridade tem de ser mantida;
- É necessário evitar movimentação dos dados do utilizador pois alguns são de caráter confidencial e não podem ser retirados do ambiente do cliente;
- As configurações do utilizador têm de ser mantidas ou, pelo menos, possíveis de ser recuperadas;

- Não é possível passar do *Etch* para o *Wheezy* sem fazer as respetivas passagens intermédias, *Lenny, Squeeze*, a não ser que se reescreva completamente o disco;
- Tem de haver espaço livre suficiente para a nova versão ser suportada;
- Não é possível aumentar o espaço disponível no que toca à *Cloud*;
- O custo de operação ou de adquirir uma *IPBrick* não pode aumentar;
- O processamento destes *updates* ou espaço necessário para os fazer não pode depender da empresa;
- Utilizador não pode sofrer paragem dos serviços que está a utilizar;
- Utilizador não tem de ser obrigado a reiniciar a ou as máquinas para as configurações e/ou atualizações surtirem efeito.

3.1.2 Cliente - Repositório

Esta solução baseia-se numa ligação cliente/servidor sendo que o modo de funcionamento é relativamente simples, trata-se do lado do cliente da extração das informações necessárias para poder passar ao servidor e este poder enviar os *updates* disponíveis, do lado do servidor trata-se de fazer a correspondência das informações dos clientes com os *updates* disponíveis para estes.

A Figura 3.1 demonstra o funcionamento que seria previsto a nível macroscópico da interação entre o cliente e o repositório, sendo que o *fetch* do lado do cliente trata-se de ir buscar as informações para passar ao servidor e o *fetch* do lado do repositório trata-se de procurar os *updates* que o utilizador tem disponível consoante a sua versão, a sua última atualização instalada, as suas aplicações e se pretende efetivamente fazer o *download* ou não (depende das definições que foram estabelecidas no cliente).

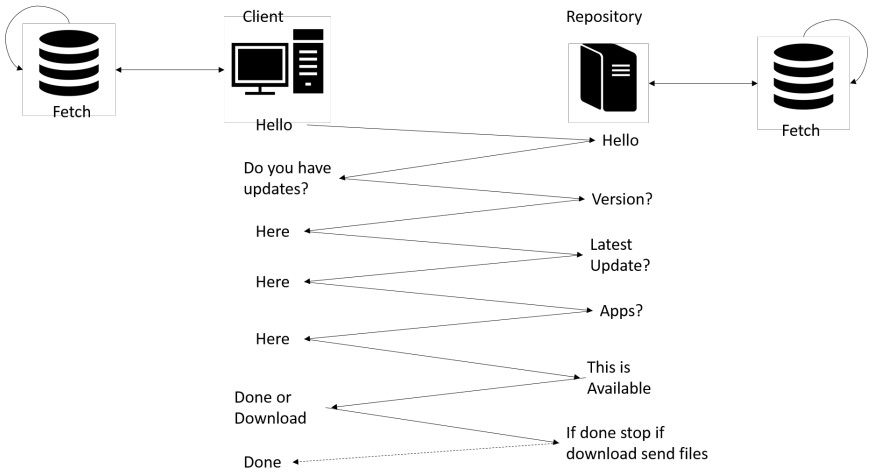

Figura 3.1: Modelo de Funcionamento da Comunicação entre o Repositório e o Cliente.

Do lado do cliente as informações necessárias seriam:

- Chave de Produto
- Versão do *IPBrick OS*
- Último *Update* instalado
- Aplicações Instaladas
- Versões das Aplicações
- Verificar *MD5SUM* para garantir que todos os ficheiros estão corretos
- Sendo mais do que um *minor update* possível introduzi-los todos em simultâneo.

Funções do lado do servidor:

- Análise da validade da Chave do Produto
- Verificar se existem *Major Updates*
- Verificar se existem *Minor Updates*
- Verificar as atualizações existentes para as aplicações
- Enviar *MD5SUM* juntamente com os ficheiros

3.1.2.1 Ficheiros Ordenados

Uma das formas de conseguir o funcionamento correto e *download* do Módulo é organizando o repositório por ordem decrescente de forma a que as atualizações mais recentes sejam as primeiras a aparecer numa verificação como um sistema de *array*, logo aquando de um *download* o repositório terá de pesquisar todos os ficheiros de cada pasta um a um para enviar todos os *updates* que o utilizador ainda não tenha instalado, a Figura 3.2 mostra um esquema de como estará organizado o repositório através desta solução e qual é o encadeamento a ser feito.

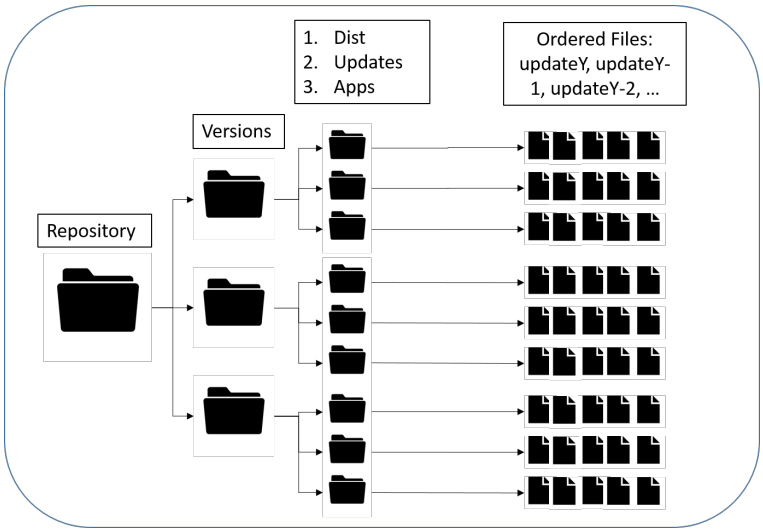

Figura 3.2: *Fetch* dos ficheiros para *update*.

3.1.2.2 *XML Cache*

Na Secção 2.6.2.2 foi descrito que o *XML* permite impor regras ao funcionamento de uma certa máquina com recurso às *markups* desta linguagem, ou seja, consegue moldar-se o seu comportamento ou mapear os dados. O objetivo desta integração do repositório com um ficheiro *XML* é poupar recursos, uma vez que se diminui o tempo de acesso (não é necessário procurar todos os ficheiros uma vez que já estão mapeados), e diminuir o tempo de resposta do repositório e consequentemente o tempo necessário de *download* uma vez que os ficheiros são enviados todos de uma vez[1].

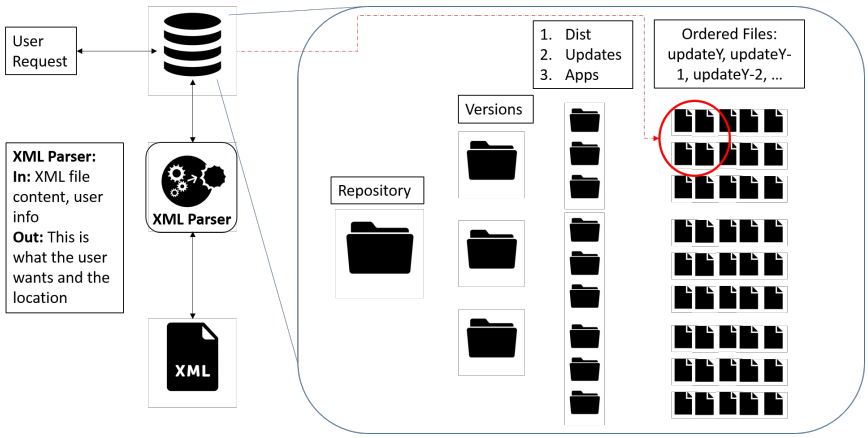

Figura 3.3: *Fetch* dos ficheiros para *update* com ficheiro *XML* como intermediário.

[1]O funcionamento é semelhante a um *proxy* que filtra a informação e diz ao *host* a informação pretendida.

3.1.2.3 Etapas da Solução

1. Desenvolver método para ser possível atualizar da *IPBrick-vX.Y* para *IPBrick-vZ.Y*;
2. Desenvolver Algoritmo para automatismo das atualizações *IPBrick-vX.Y* para *IPBrick-vX.Z*;
3. Desenvolver Interface Gráfica;
4. Estabelecer regras de mapeamento e organização para o ficheiro *XML*.
5. Desenvolver Algoritmo de comunicação Cliente -> Servidor;
6. Desenvolver Algoritmo de comunicação Servidor -> Cliente;

3.2 Interface Gráfica *WEB*

3.2.1 Opções Disponíveis

As opções disponíveis não são mais do que a capacidade de personalização do modo como os *updates* são feitos que a *GUI* deve mostrar e permitir editar. Estes requisitos constituem:

1. Fazer atualizações automáticas imediatamente para versão mais recente independentemente dos programas que estão a ser executados;
2. Fazer automaticamente atualizações onde não for preciso reiniciar o sistema;
3. Permitir agendar *updates* para uma certa data e hora;
4. Escolher a periodicidade com que o sistema procura por atualizações (predefinido = 1 mês);
5. Permitir escolher que serviços podem ser parados aquando de uma atualização;
6. Permitir ligar e desligar notificações de novas atualizações;
7. Ver as notas sobre uma determinada atualização;
8. Opção de nunca reiniciar sem perguntar ao utilizador primeiro.

3.2.2 Protótipo e seu Funcionamento

Aquando da entrada no *IPBrick OS* selecionando a opção *Advanced Configurations* e nesse menu *IPBrick* obtém-se um outro menu onde uma das opções é *Updates*, tal como é representado na Figura 3.4.

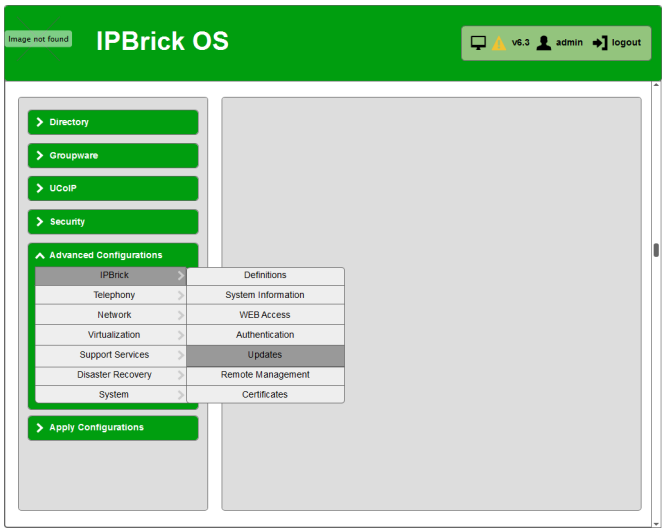

Figura 3.4: Entrada *Updates IPBrick v6.3*.

Após entrada na opção *Updates* o menu que tinha surgido desaparece e tudo o resto mantém-se sendo que aparecem as opções para realizar atualizações como é demonstrado na Figura 3.5. Nesta página a primeira tabela demonstra as atualizações que já foram introduzidas através de um visto e as que ainda não foram com uma *checkbox* por selecionar bem como um *link* para poder aceder ao histórico, *History*, visível através da Figura 3.6.

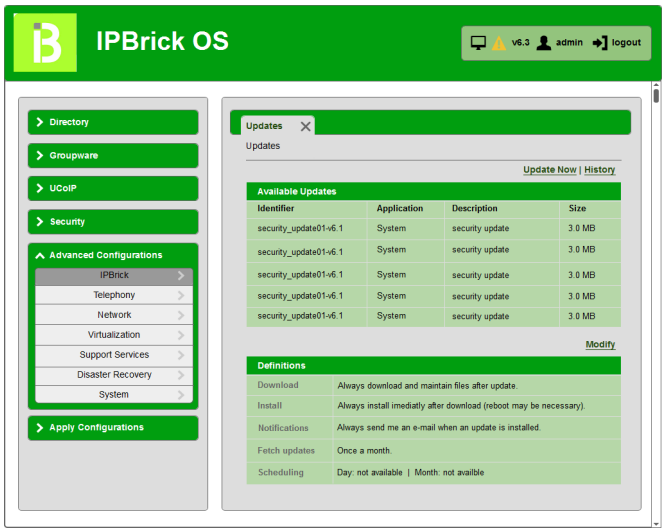

Figura 3.5: Página de Atualizações.

Figura 3.6: Histórico de atualizações feitas.

Ainda dentro do menu principal é possível ver a tabela que contém as opções predefinidas. Carregando no *link "Definitons"* abre uma nova janela onde será possível escolher que tipo de automatismos e opções poderão ser utilizadas. Está previsto também a existência de uma agenda onde o utilizador escolhe a que horas e dias pretende que as atualizações sejam feitas e com que periodicidade são procuradas novas atualizações, a visão geral destas opções está disponível na Figura 3.7.

Figura 3.7: Opções disponíveis para fazer atualizações.

Dentro das opções para *Download*, Figura 3.8, será possível controlar, gerir e decidir sobre como o módulo lidará com os ficheiros resultantes desta operação. Por outras palavras, permite

escolher: Se o descarregamento é feito e os ficheiros ficam em sistema até a instalação da atualização ser efetuada ou então se são imediatamente apagados após o processo. O utilizador pode ainda escolher a opção de nunca efetuar os *downloads*.

Figura 3.8: Opções de *Download*.

Este protótipo, Figura 3.9 pretende mostrar que opções de periodicidade de pesquisa de *updates* estão disponíveis.

Figura 3.9: Opções de pesquisa de *updates*.

Quanto à instalação das atualizações, Figura 3.10, será possível gerir em que data e hora é que a instalação será feita (independentemente de parar serviços), se é aceitável ou não parar serviços na altura do *download* ou então se simplesmente não se pretende fazer de forma alguma a atualização.

Figura 3.10: Opções de instalação.

Por fim falta apenas mostrar de que modo é possível agendar a instalação dos *updates*, Figuras 3.11, 3.12 e 3.13. Os parâmetros são dia, mês e hora. O ano não aparece porque para além de fazer pouco sentido a nível de instalação, o próprio sistema sabe a data presente portanto se na altura do agendamento a data for 15-12-2016 e o agendamento estiver marcado para 14-12, o sistema sabe que será apenas no ano seguinte. O campo hora e minutos caso sejam mantidos no *default* o módulo fara a atualização a qualquer hora do dia indicado.

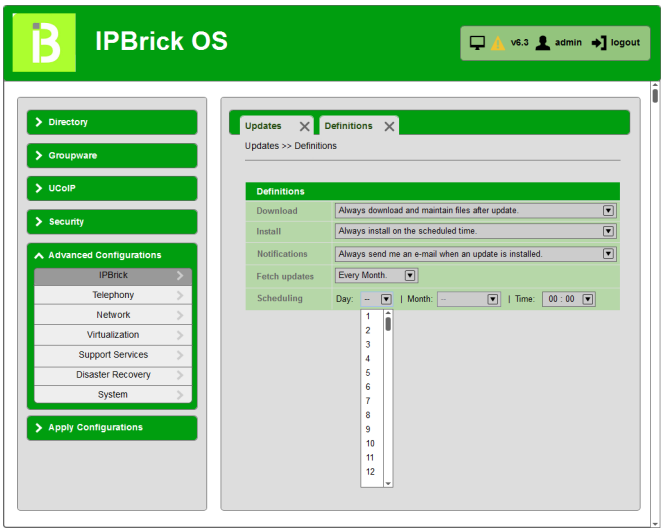

Figura 3.11: Opções de agendamento: dia.

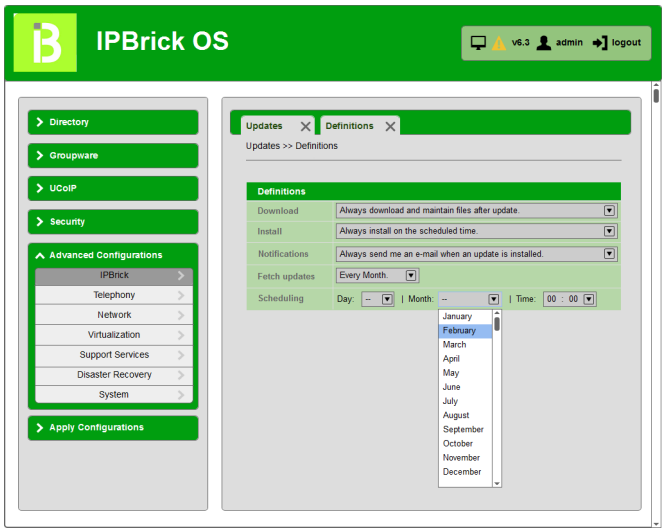

Figura 3.12: Opções de agendamento: mês.

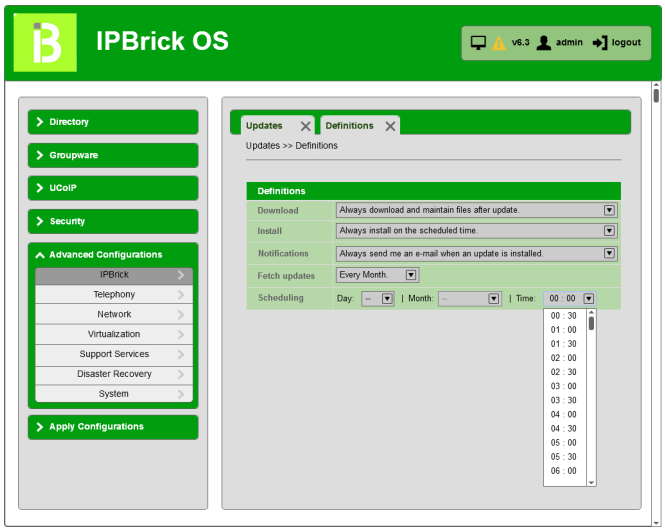

Figura 3.13: Opções de agendamento: relógio.

As notificações são enviadas para o *e-mail* do cliente e podem ser para avisar que existe um novo *update* disponível e que foi instalado um novo *update*. Pode ainda ser avisado dos dois eventos anteriores ou então pode preferir não ser avisado de forma alguma, Figura 3.14.

Figura 3.14: Avisos relativamente às atualizações.

3.3 *Major Updates*

Uma das soluções que o *software IPBrick* precisava era ao nível das atualizações de uma distribuição para outra, mais precisamente da *IPBrick v5.x* para uma *IPBrick v6.x*. Ao nível do *Linux Debian* isto trata-se de um salto do *Etch* para o *Wheezy*, ou seja, do *Debian v4.x* para a *v7.x*.

3.3.1 Soluções Descartadas

Tendo em conta os requisitos presentes no Capítulo 3.1.1 grande parte das soluções inicialmente pensadas e propostas ficaram postas de parte, nomeadamente:

- Mecanismos semelhantes a *Live Migration*;
- Servidores *IPBrick* de suporte a *updates*.
- Contratar mais espaço nas máquinas presentes na *Cloud*;

3.3.1.1 *Live Migration*

Baseando-se nos algoritmos de *LM* presentes na secção 2.3.2 e tendo em conta que do ponto de vista teórico esta seria a opção mais vantajosa visto que gastaria menos recursos num certo período de tempo e seria aparentemente mais rápida, concebeu-se a ideia de uma solução baseada no funcionamento de *LM*.

O processo começaria por lançar-se uma *VM*, *maq2* com a última versão do *software* enquanto a outra, *maq1*, se encontra em funcionamento. Após isto iniciar-se-ia o processo de passagem de dados da primeira máquina para a segunda, que irá manter-se durante todo o processo até o

funcionamento da *maq1* ser interrompido. De seguida, são copiadas as páginas que estão a ser visualizadas pelo utilizador e redirecionar-se-ia o tráfico da máquina original e por fim sincronizar-se todos os restantes elementos. *A posteriori* a segunda máquina resumiria as funções e a primeira seria apagada. Todo este processo tem de ser supervisionado por uma terceira máquina para garantir que tudo corre como o previsto. O esquema da Figura 3.15, em baixo, permite uma melhor perceção do respetivo processo.

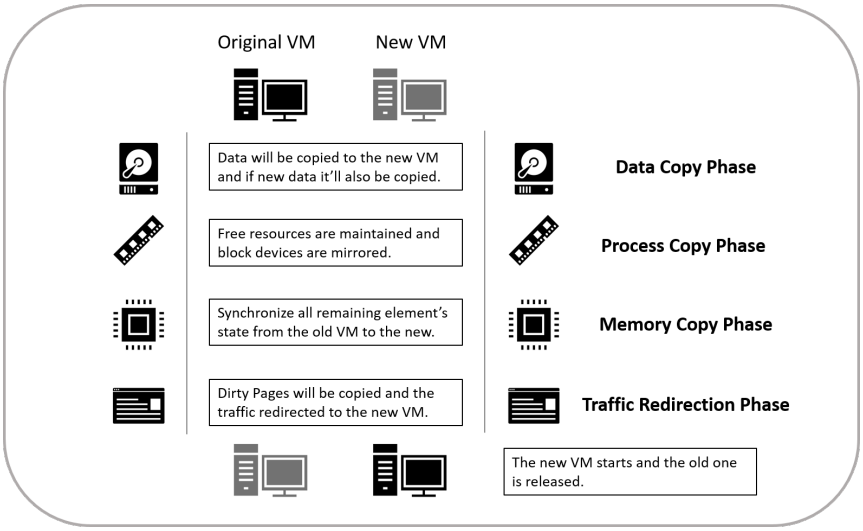

Figura 3.15: Algoritmo baseado em *LM*.

Este processo traria grande valor de mercado principalmente porque, na perspetiva do utilizador, o sistema nunca pararia de trabalhar, simplesmente sofreria um *update*, ou seja, num momento apresentava-se de uma forma e no outro, após um *refresh* de página, apresentar-se-ia de outra forma. Não haveria propriamente interrupção de longa duração dos serviços que o cliente estivesse a utilizar e não haveria a necessidade de reiniciar a máquina e, para além deste valor acrescentado permitia também que atualizações que normalmente demorariam muito tempo passassem a ser muito mais rápidas.

Por oposição este método teria algumas desvantagens e analisando as limitações impostas pelo sistema e pela empresa consegue-se retirar os seguintes pontos:

- Em primeiro lugar ao nível das aplicações e processos, uma aplicação estar presente numa versão não quer dizer necessariamente que esteja presente na seguinte;

- Em segundo lugar, ao nível do sistema de ficheiros, da localização dos ficheiros ou de funções existentes, não há garantias que uma função ou um *package* tenham sido mantidos constantes o que pode levar a que na versão anterior se use a função x e na versão atual se use a função y que até pode utilizar a função x mas como o sistema estava preparado para x provocar um *crash*;

- Em terceiro lugar, seria necessário recorrer a intervenientes externos ou *APIs* externas, como a da *IBM* onde a *IPBrick* tem algumas das suas máquinas virtuais, para conseguir criar um processo automatizado;

- Em último lugar, são necessárias duas máquinas extra para fazer o *update* o que aumenta os custos da Empresa não só pela necessidade de as lançar mas também porque podem ser necessárias melhorias em termos da virtualização de recursos físicos, ou seja, memória, processador, disco rígido e etc.

3.3.1.2 Servidores de Suporte

Esta ideia baseava-se em utilizar os servidores da *IPBrick* como sistema de apoio ao *update*, ou seja, todos os dados que o cliente tivesse eram transferidos para um servidor externo onde eram armazenados até a operação ficar concluída. O processo seria relativamente simples como se pode verificar na Figura 3.16 seguiria os seguintes passos:

1. *Download*;
2. Descompactar;
3. Instalar [2];
4. Repor configurações;
5. Repor Dados;
6. Reiniciar.

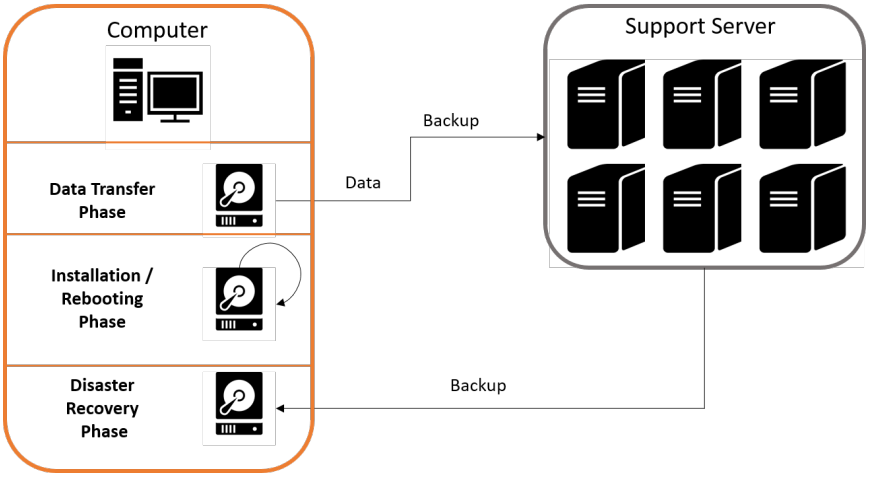

Figura 3.16: Esquema de funcionamento "Servidor de Suporte".

[2]Este passo implica que o disco seja todo reescrito e portanto todos os dados apagados.

Vantagens	Desvantagens.
Despreocupação com espaço da parte do utilizador.	Necessário ter um servidor constantemente disponível.
Integridade dos dados garantida[3].	Gastos extras graças à necessidade de servidores.
Apenas necessita de uma ligação à *Internet* para realizar o processo.	Informação sensível está momentaneamente fora da empresa.
Possível tanto para máquinas *indoor* ou *cloud*.	Necessidade de muitos recursos (processamento, memória e capacidade).

Tabela 3.1: Vantagens e desvantagens da utilização da solução "Servidores de Suporte".

Rapidamente através da análise da Tabela 3.1 e das limitações existentes - Secção 3.1.1 - verifica-se que esta solução não era nem é viável neste momento e, por isso, foi descartada.

3.3.2 Soluções Consideradas

As soluções consideradas são as que da *pool* de ideias, depois de se ter chegado à conclusão que cumprir todos seria algo impossível sem aumentar os custos, obedecem aos critérios mínimos estabelecidos:

1. Respeitar o maior número de limitações empresariais possíveis.
2. Não gerar qualquer tipo de custo adicional.

Nesse sentido conseguiu escolher-se duas soluções que estão descritas de seguida.

3.3.2.1 *Chroot Bypass*

A versão 5.3 da *IPBrick* é construída sobre o *Linux Debian Etch* e a versão 6.1 sobre *Linux Debian Wheezy* para fazer este salto, normalmente, seria necessário passar para o *Lenny*, de seguida para o *Squeeze* e depois para o *Wheezy* sendo que em cada uma destas passagens existe a necessidade de apagar módulos e inserir módulos o que torna o processo bastante complexo.

Este processo torna-se ainda mais complexo quando se tem de cumprir obrigatoriamente uma das imagens de marca da empresa: manter todos os dados intactos, manter as configurações escolhidas pelo *user* e o processo ser relativamente rápido ou que pareça rápido para o utilizador (utilizando processos em segundo plano).

Esta solução tem como suporte as características do ambiente *chroot* e do *Hard Link* referenciadas na Secção 2.1.3.1 do Capítulo 2.1 e baseia-se na ideia de ter uma de ligação entre a base do diretório árvore, */*, e uma pasta dentro do *chroot* de forma a que mesmo estando dentro da *jail* seja possível ter acesso ao sistema que se pretende alterar. A ideia pretende criar um ambiente minimalista onde teria acesso ao sistema principal através de um *link*, Secção 2.1.3.2, partindo desta ligação as partições */, /usr, /var, /opt* serão formatadas e convertidas para *EXT4* para que no

passo seguinte se possa descompactar os *drives* necessários para o sistema funcionar, para uma melhor compreensão ver Figura 3.17.

Espera-se que desta forma seja possível transformar uma *IPBrick-v5.3* na versão 6.1, sem ser necessário reestruturar o que já existe tal como são as normas recomendadas para a introdução de uma nova tecnologia.

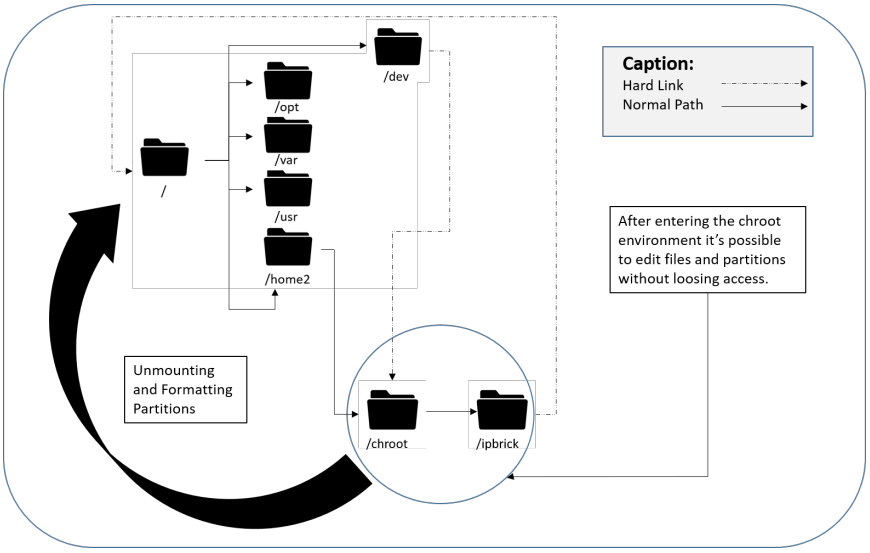

Figura 3.17: Esquema de funcionamento *Chroot Bypass*.

3.3.2.2 Sincronizar, Reestruturar, Reescrever, Descompactar e Reiniciar(*SRRDR*)

A sincronização aparece logo no início do processo porque, tal como foi descrito na secção 2.4.2.1, o *IPBrick OS* tem um esquema de partições muito particular na versão 5.3 e uma vez que será necessário fazer uma reestruturação do esquema no passo seguinte será preciso que os conteúdos existentes nas partições que irão desaparecer sejam guardados ou passíveis de ser recuperados ou acedidos de outra forma. Aqui entra uma ferramenta importante para este processo, o comando *rsync* [71].

O *rsync* permite a sincronização de ficheiros entre dois pontos, o algoritmo deteta as similaridades e depois copia a diferença. Este comando será utilizado para aproveitar o conteúdo das pastas onde estão montadas as partições *usr, var, opt* e coloca-lo-á numa pasta dentro de uma das partições que não serão alteradas (*home1* e *home2*) garantindo assim que aquando da remoção das pastas, respetivas partições e funções contidas nestas e necessárias para executar outros comandos estejam disponíveis através da implementação de *soft links*, ver secção 2.1.3.2.

Este método envolve obrigatoriamente uma reestruturação do esquema de partições e redimensionamento das mesmas sendo que o objetivo é a alocação de todo o espaço livre numa única partição para que essa partição tenha o espaço livre suficiente para fazer o *update*. Esta reestruturação iniciar-se-á na remoção das partições correspondentes às pastas descritas anteriormente e depois passará pelo redimensionamento da partição secundária, ou seja, passar o ponto de início

da partição para o ponto de inicio da partição correspondente à pasta *home1*. Desta forma todo o espaço que fica livre para resulta na criação de uma nova partição que servirá como novo /, novo diretório *root*.

Nesta partição é onde habitará a nova versão funcional enquanto que na partição onde residia a versão anterior será transformada numa partição dedicada única e exclusivamente a atualizações para garantir que existe sempre um modo de os fazer e espaço suficiente para o efeito. Desta forma garante-se a integridade dos dados do utilizador e guardando as configurações na *homeY* permite que sejam convertidas para serem compatíveis com a nova versão podendo portanto ser recuperadas e implementadas na nova versão.

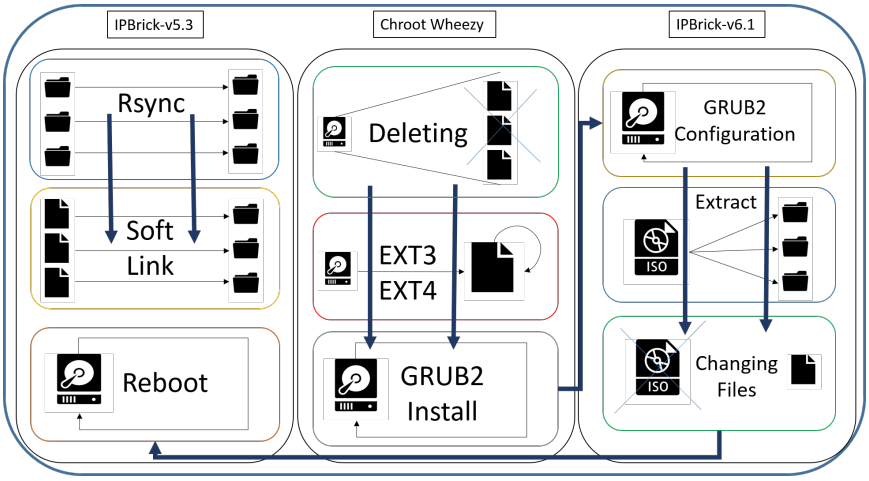

Figura 3.18: Esquema de funcionamento *SRRD*.

3.3.3 Etapas de Solução

Nesta Secção encontram-se as etapas que são precisas para conseguir realizar o *update* com cada uma das soluções.

3.3.3.1 *Chroot Bypass*

Processo idealizado de acordo com a Secção 3.3.2.1 tem as seguintes etapas:

1. Instalar o *Debootstrap*
2. Criar as pastas necessárias dentro das partições que não vão ser alteradas - *chroot, newroot, opt, var, usr, ipbrick*
3. Instalar o ambiente *Wheezy* minimalista
4. Sincronizar a pasta */dev* do sistema original com a mesma pasta do sistema minimalista
5. Fazer um *Hard Link* de / para a pasta *ipbrick*

6. Entrar no *chroot*
7. Formatar as partições */, usr, var, opt*
8. Fazer novos sistemas de ficheiros, de preferência *EXT4*
9. Fazer as alterações necessárias aos ficheiros de arranque, por exemplo: */etc/fstab*
10. Descompactar para as partições formatadas os ficheiros *driveY.dat* com as permissões corretas
11. Instalar o *GRUB2*
12. Reiniciar e verificar integridade do sistema
13. Migrar as configurações e aplicar *Disaster Recovery* para recuperar as configurações do utilizador.

3.3.3.2 Sincronizar, Reestruturar, Reescrever, Descompactar e Reiniciar (*SRRDR*)

1. Instalar o *Debootstrap*
2. Criar as pastas necessárias dentro das partições que não vão ser alteradas - */mychrootdir, /newroot, /mylinkdir, /mysyncdir*
3. Instalar o ambiente *Wheezy* minimalista
4. *Hard Link /dev* para a mesma pasta do sistema minimalista
5. Sincronizar através de *rsync* as pastas */usr, /var, /opt* com */mysyncdir*
6. Apagar as pastas sincronizadas e *Soft link* com as correspondentes em */mysyncdir*
7. Entrar no *chroot*
8. Apagar as partições *usr, var, opt*
9. Redimensionar a partição lógica retirando-lhe o espaço deixado por */var, /opt* e criar uma nova partição primária (*EXT3* ou *EXT4*, se possível) com esse e com o espaço de */usr*.
10. Introduzir e descompactar os ficheiros *drive.dat* na nova partição.
11. Instalar o *GRUB2* em */mylinkdir/*, a partir do *chroot*, ou seja, em */newroot/*.
12. Configurar o *GRUB2* e guardar o ficheiro *grub.cfg* em */mylinkdir/boot/grub/*.
13. Fazer as alterações necessárias aos ficheiros de arranque: */fstab* e */grub.cfg*.
14. Reiniciar e verificar integridade do sistema

3.4 *Minor Updates*

3.4.1 Requisitos

Nas atualizações dentro da mesma versão é necessário corresponder a alguns requisitos para o bom funcionamento do módulo mas sobretudo para haver a capacidade de corresponder a todas as opções que foram propostas, principalmente as que não envolvem que haja *reboot* ou paragem de serviços em funcionamento. Posto isto, retira-se que para o algoritmo funcionar e para se conseguir implementar todas as opções referidas, secção 3.2.1, é necessário conseguir fazer os seguintes requisitos:

- Obter a versão que está a ser utilizada;
- Obter o último *update* instalado;
- Obter as atualizações disponíveis;
- Obter o tamanho do próximo *update*;
- Guardar e obter sempre que necessário as configurações de *update* utilizadas;
- Notificar o utilizador da existência de novas atualizações caso seja essa a opção escolhida por ele;
- Notificar a necessidade de reiniciar o sistema;
- Notificar a necessidade de parar serviços temporariamente;
- Listar todas as atualizações instaladas e respetivas informações;
- Agendar a instalação de instalações;
- Todas as máquinas recebem aviso de nova instalação disponível (*Server Side*);
- Permitir ao utilizador selecionar atualizações;
- Notificar utilizador da recursividade inerente a uma atualização selecionada;

3.4.2 Nomenclatura das Atualizações

Neste momento as atualizações lançadas dentro da mesma distribuição podem ter os seguintes nomes:

- *Fix Updates*
- *Security Updates*
- *Packages for Updates*

- *Updates*

Como as próprias traduções indicam, os primeiros tratam-se de atualizações para comatar uma falha ou um erro, os segundos tratam-se de melhorias de segurança, os terceiros são pacotes de funções necessárias para se conseguir fazer uma atualização e por último temos efetivamente as verdadeiras atualizações.

No entanto, do ponto de vista do automatismo, as várias diferenças na nomenclatura dificultam o processo o que implicará uma supressão da primeira e terceira nomenclatura.

3.4.3 Solução

A solução nas atualizações de segurança, correção de erros, melhoria de *features* ou de acrescentar funções têm já um sistema desenvolvido, Secção 2.5.3, falta apenas um automatismo que será garantido pelo próprio módulo.

Tendo em conta a quantidade de atualizações por versão da *IPBrick* e o tempo demorado em geral, a grande melhoria que esta solução vai proporcionar é o conforto do utilizador e melhorar a eficiência. Isto é conseguido não só pelo automatismo em si mas também pelo facto de impedir enganos na recursividade.

Por outro lado, é suposto o módulo permitir a instalação de vários *updates* em simultâneo caso exista mais do que um e caso seja essa a opção do utilizador, uma vez que o *Linux Debian* permite essa possibilidade através do comando *apt-get* embora seja o utilizador a definir se será possível fazer a atualização dessa forma (Secção 3.2.1 esclarece melhor o controlo do utilizador), o que vai permitir que os *updates* se tornem também mais eficientes, menos tempo por não haver paragens e portanto menos tempo de serviços indisponíveis.

O procedimento na instalação destas atualizações vai passar por receber os ficheiros do servidor sempre que estiverem disponíveis após pedido e sendo permitido pelo *user* e verificar qual é a ordem de instalação procedendo à respetiva introdução no sistema. Esta introdução no sistema está dependente das opções escolhidas pelo utilizador.

Caso o utilizador selecione a opção de impedir a atualização enquanto o processo está em execução, por exemplo, o algoritmo irá impor um *sleep* até o processo ser destruído. Quando este é finalizado gera uma interrupção que permite continuar com a atualização.

3.5 Conclusão

Neste Capitulo foram apresentadas as limitações impostas pelo sistema operativo *IPBrick* que estão diretamente relacionadas com as limitações impostas pelo *Linux Debian Etch* e a sua passagem para três versões superiores sem recursividade. Referenciou-se também os requisitos e limitações impostas pela própria Empresa como barreiras que não podiam ser ultrapassadas e objetivos que tinham de ser garantidos. Foram apresentadas e explicadas as ideias que podiam resultar no Módulo pretendido que sendo divido em três grandes partes foram apresentadas soluções para todas ou não havendo propriamente uma solução a devida justificação para tal.

Posto o acima descrito e través do cruzamento das informações, descreveu-se como irá funcionar o Módulo para Gestão de Atualizações de *Software*, nomeadamente como vão ser obtidas as novas atualizações, sob que contextos e que regras são aplicadas e que tipo de influência o utilizador e a Empresa têm sobre a forma como o sistema operativo se comporta, isto a nível macroscópico. A implementação do algoritmo deste módulo vai ser feita com recurso a *PHP*, do lado do cliente, para que possa ser integrado juntamente com *HTML, JavaScript* e *CSS* da interface gráfica e com *Bash* para instalação dos *updates*. Do lado do servidor será necessário utilizar *XML* para tornar o processo mais eficiente, *SQL* para extrair as atualizações disponíveis do repositório e por fim o protocolo *TCP/IP* para a transferência e comunicação com o cliente que será implementado também através de *PHP*.

Ao nível específico e dentro das atualizações de distribuição foram apresentadas quatro soluções. Duas das quais foram consideradas e serão testadas com recurso a máquinas virtuais fornecidas pela Empresa como será passível de verificar no Capitulo seguinte. Para estas soluções serão necessários conhecimentos de *Bash*, sistema de ficheiros *Linux*, *GRUB* e gestão de partições. Também se encontram descritas duas outras soluções que forma descartadas devido a haver conflitos com as limitações de alta prioridade, todas as razões e conflitos estão explicados.

Dentro das atualizações "pequenas"foi explicado que já existe uma forma de as aplicar e instalar, só não existe o automatismo necessário para comodidade e facilidade do utilizador e que acrescenta valor à Empresa. Explica-se também que este automatismo, comodidade e rapidez nas atualizações será conseguida não só pelo paralelismo disponível aquando da existência de várias atualizações pendentes mas também será conseguido na própria construção do algoritmo de funcionamento do *MGAS*, descrito na primeira Secção deste Capitulo.

No que toca a interface gráfica, é apresentado todo o *design* idealizado através de imagens protótipo e da explicação de cada uma das imagens referenciado as opções permitidas ao utilizador escolher. Foram apresentadas todos os cenários possíveis que qualquer cliente consegue obter.

Capítulo 4

Implementações e Testes

4.1 Testes

4.1.1 *Major Updates*

Para se conseguir ter uma perceção completa de como as atualizações são feitas foi necessário ter a experiência que qualquer utilizador do *IPBrick OS* tem. Nesse sentido, começou-se por realizar os *major updates* da forma que é possível executa-los neste momento, reescrevendo o disco.

Todos os testes ocorreram em máquinas virtuais com as mesmas características:

- 1024 *MB* de *RAM*;
- 1 processador à frequência de 2.4 *GHz*

4.1.1.1 Espaço Necessário

Um dos maiores problemas para a integração do Módulo de Gestão de Atualizações trata-se do espaço, ou seja, como conseguir fazer as atualizações na *Cloud* onde o espaço "não pode"ser alterado. Neste sentido, o primeiro teste que qualquer instalação terá de fazer é saber qual é o espaço mínimo necessário para fazer o *update* e este primeiro teste permite verificar isso mesmo, que diferenças existem entre as versões em termos de sistema, aplicações e espaço final ocupado. A Tabela 4.1 apresenta não só a informação pretendida mas também tem em conta outros blocos importantes e em que é que isso resulta em termos de espaço final.

Espaço Ocupado		
Pastas[1]	*IPBrick-v5.3*	*IPBrick-v6.1*
bin	*3.7 MB*	*6.1 MB*
etc	*34 MB*	*47 MB*
boot	*20 MB*	*27 M*
lib	*138 MB*	*226 MB*
dev	*85 KB*	*92 KB*
opt	*218 MB*	*621 MB*
/	*284 KB*	*8.5 MB*
Total Apurado:	*3 GB*[2]	*6 GB*[2]

Tabela 4.1: Valores retirados de máquinas com todas as instalações feitas.

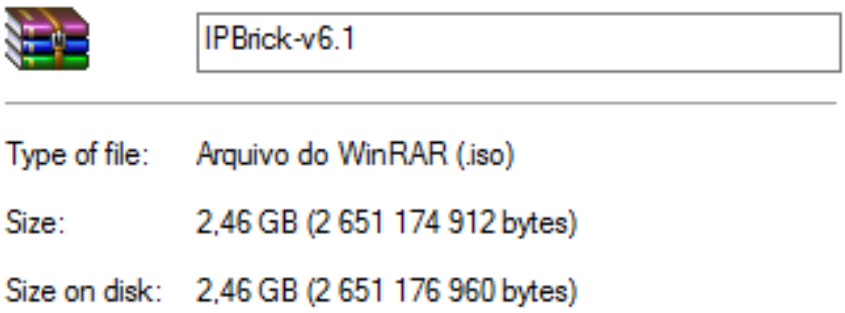

Figura 4.1: *Espaço ocupado pela ISO da IPBrick-v6.1.*

Tendo em conta a tabela acima e a Figura 4.1 consegue-se facilmente fazer o cálculo:

$$Espaco = 3GB(IPB\text{-}5.3) + 6GB(IPB\text{-}6.1) + 2.5GB(ISO) + 2.5GB(Extract) + 1GB(chroot) = 15GB \tag{4.1}$$

4.1.1.2 Instaladores *IPBrick*

Os primeiros testes consistiram na instalação de uma *IPBrick-v5.3* numa máquina virtual e testar o funcionamento das atualizações para a versão seguinte e depois ao retroceder a atualização também através de uma ISO. O resultado foi que ao fazer a instalação por cima da versão instalada o *software* de instalação é capaz de reconhecer esta versão anterior e pergunta ao utilizador se pretende continuar com a instalação, Figura 4.2.

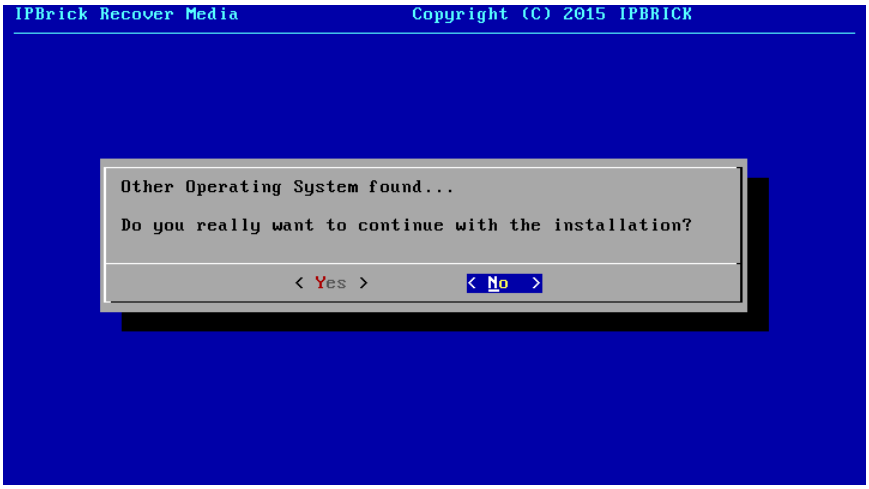

Figura 4.2: Reconhecimento da versão anterior.

Por oposição quando se faz o inverso, introduzir uma *ISO* de uma versão anterior, a instalação é feita como se não houvesse sistema nenhum instalado e o disco é formatado.

4.1.1.3 *Chroot Bypass*

Esta solução baseando-se nas propriedades do *chroot* e dos *hard links* torna-se preponderante testa-los e perceber se efetivamente é possível aceder ao sistema e manipula-lo através do ambiente minimalista, como tal a primeira etapa foi a instalação do *chroot* através do *debootstrap*, tal como apresentado na secção 3.3.2.1 do Capitulo 3.

Aqui surge o primeiro problema, o *Debian Etch* por ser uma versão antiga não possui os *scripts* para instalação dos ambientes minimalistas acima, como o *lenny* ou *wheezy*, por isso foi necessário passar todos os ficheiros do ambiente por *SSH*, *Secure SHell*, para uma pasta da máquina virtual a ser utilizada para testes, Figura 4.3.

```
ipbrick:/oldroot#
ipbrick:/oldroot#
ipbrick:/oldroot# chroot /home2/debootstrap/i386/
root@ipbrick:/#
root@ipbrick:/#
root@ipbrick:/# ls -l
total 125
drwxr-xr-x  4 root root   1024 Apr 11 08:36 --verbose
drwxr-xr-x  2 root root   2048 Apr 12 09:13 bin
drwxr-xr-x  3 root root   1024 Apr  6 05:32 boot
drwxr-xr-x  3 root root   1024 Mar 28 10:49 dev
drwxr-xr-x 50 root root   2048 Apr 13 17:17 etc
drwxr-xr-x  2 root root   1024 Mar 28 10:50 home
drwxr-xr-x  2 root root   1024 Apr 11 01:19 ipbrick
drwxr-xr-x 12 root root   1024 Apr 12 11:06 lib
drwxr-xr-x  2 root root   1024 Mar 28 10:49 media
drwxr-xr-x  2 root root   1024 Mar 28 10:49 mnt
drwxr-xr-x  2 root root   1024 Mar 28 10:49 opt
drwxr-xr-x  3 root root   1024 Apr  6 05:27 proc
drwx------  2 root root   1024 Apr 12 11:05 root
drwxr-xr-x  7 root root   1024 Apr 12 12:39 run
drwxr-xr-x  2 root root   3072 Apr 11 00:28 sbin
drwxr-xr-x  2 root root   1024 Mar 28 10:49 selinux
drwxr-xr-x  2 root root   1024 Mar 28 10:49 srv
drwxr-xr-x  2 root root   1024 Mar 28 10:49 sys
drwxr-xr-t  2 root root   1024 Apr 13 17:17 tmp
-rw-r--r--  1 root root 101376 Apr 12 12:27 typescript
drwxr-xr-x 10 root root   1024 Mar 28 10:49 usr
drwxr-xr-x 11 root root   1024 Mar 28 10:50 var
root@ipbrick:/#
root@ipbrick:/#
root@ipbrick:/#
```

Figura 4.3: Listagem dos ficheiros necessários para um ambiente minimalista ser utilizado.

De seguida era necessário estabelecer a ligação entre o sistema e o *chroot* para isso foi testado o funcionamento dos*hard links* e dos *softlinks*, Secção 2.1.3.2 e 2.1.3.2, bem como como é que o sistema se comporta após *reboot*. Apenas com os *HL* se conseguiu obter acesso a todas as pastas que se pretendia que fossem substituídas */, /user, /opt, /var*. Com recurso a estas definições iniciais alterou-se o ficheiro responsável pela montagem inicial das partições para que em qualquer momento onde fosse necessário *reboot* o *OS* soubesse onde teria de pôr os pontos de ligação para se poder continuar a operação, a Figura 4.4 permitem ver o ficheiro inicial.

```
ipbrick:/#
ipbrick:/#
ipbrick:/# cat /etc/fstab
# /etc/fstab: static file system information.
#
# <file system>         <mount point>   <type>  <options>       <dump>  <pass>
proc                    /proc           proc    defaults                0       0
UUID=65dcec43-9194-4f53-aee6-2cab98d3dba7       /       ext3    errors=remount-ro       0       1
UUID=e7ba47ef-4224-4a6a-ae2f-017ecc3f2623       none    swap    sw      0       0
/dev/fd0                /floppy         auto    user,noauto             0       0
/dev/cdrom              /cdrom          iso9660 ro,user,noauto          0       0
UUID=a0db1a3b-d141-40c6-a8e2-ddd5f181e54d       /usr    ext3    defaults        0       2
UUID=0648294a-2ebd-4581-bbb3-488aa565dd4c       /var    ext3    defaults        0       2
UUID=b21f5b40-ba6d-4194-bfe8-8cb81a2325ce       /opt    ext3    defaults        0       2
UUID=10978567-2a26-48d0-a3f4-2eb44222a288               /home1  ext3    defaults,usrquota,grpquota,user_xattr,acl       0       2
UUID=b11414fb-cc38-4239-86c5-1ab4c08c5251               /home2  ext3    defaults,usrquota,grpquota,user_xattr,acl       0       2
LABEL=IPBRICK-D /opt/lpbox/backupDB     auto    user,noauto             0       0
/home1/_fax   /var/spool/hylafax   none     bind               0      2
ipbrick:/#
ipbrick:/#
ipbrick:/#
```

Figura 4.4: Ficheiro *fstab* original.

O esquema da Figura 2.7 e tendo em consideração a forma como *fstab* está editado na Figura 4.5 mostra um erro que foi extremamente frequente durante os testes, os *UUID* de cada partição têm apenas um *mount point* associado que se encontra dentro do *AM*, ou seja, não vão ser passível de ser acedidos no sistema original caso se pretenda fazer um *reboot* e consequentemente não vai ser possível entrar no *chroot* novamente porque o diretório */usr* não está presente com o comando para ser executado ou pelo menos o sistema não sabe onde está.

```
ipbrick:/#
ipbrick:/#
ipbrick:/# cat /oldroot/home2/fstab
# /etc/fstab: static file system information.
#
# <file system>         <mount point>   <type>  <options>       <dump>  <pass>
proc                    /proc           proc    defaults                0       0
UUID=2288ea52-6975-4a2b-9ba5-eab4bac0fddf       /home2/debootstrap/i386/ipbrick/root    ext3    errors=remount-ro       0       1
UUID=ce5617ed-a3c8-48b2-92a7-6d977bcbb1a1       none    swap    sw      0       0
/dev/fd0                /floppy         auto    user,noauto             0       0
/dev/cdrom              /cdrom          iso9660 ro,user,noauto          0       0
UUID=afb017fa-80c1-4106-a891-a505774f8742       /home2/debootstrap/i386/ipbrick/usr     ext3    defaults        0       2
UUID=0e8d7ac6-f3b7-4419-8364-5e1d148ad409       /home2/debootstrap/i386/ipbrick/var     ext3    defaults        0       2
UUID=16e94961-5380-425e-8f04-d549287620f3       /home2/debootstrap/i386/ipbrick/opt     ext3    defaults        0       2
UUID=77c190a8-168e-4ba5-9a1d-2593cf3920c7               /home1  ext3    defaults,usrquota,grpquota,user_xattr,acl       0       2
UUID=1baa4e8a-7404-43f9-aeff-ccdd7190e37e               /home2  ext3    defaults,usrquota,grpquota,user_xattr,acl       0       2
LABEL=IPBRICK-D /opt/ipbox/backupDB     auto    user,noauto             0       0
/home1/_fax    /var/spool/hylafax   none     bind                0      2
ipbrick:/#
ipbrick:/#
ipbrick:/#
```

Figura 4.5: *"Chrooted"fstab*.

Dentro do mesmo princípio ocorreram vários testes que provocaram com que esta solução fosse deixada de parte e que surgisse a segunda ideia, *SRRD*, Secção 3.3.2.2, nomeadamente na formatação das partições. Na tentativa de formatar as partições existentes surgiu um outro problema, o *Linux* não permitia a formatação[3] nem a remoção[3] da parte do disco que esteja montada[3] e em utilização, Figura 4.6.

```
ipbrick:/#
ipbrick:/# ls /dev/hda
hda   hda1  hda2  hda3  hda4  hda5  hda6  hda7  hda8
ipbrick:/# ls /dev/s
shm/     sndstat  stderr   stdin    stdout
ipbrick:/# parted
GNU Parted 1.8.8
Using /dev/hda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) p
Model: QEMU HARDDISK (ide)
Disk /dev/hda: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system  Flags
 1      32.3kB  1152MB  1152MB  primary   ext3         boot
 2      1152MB  2221MB  1069MB  primary   linux-swap
 3      2221MB  4943MB  2723MB  primary   ext3
 4      4943MB  21.5GB  16.5GB  extended
 5      4943MB  10.2GB  5289MB  logical   ext3
 6      10.2GB  11.4GB  1152MB  logical   ext3
 7      11.4GB  16.4GB  5026MB  logical   ext3
 8      16.4GB  21.5GB  5059MB  logical   ext3

(parted) rm 5
Error: Partition /dev/hda5 is being used. You must unmount it before you modify it with Parted.
(parted) quit
```

Figura 4.6: Bloqueio na remoção.

Após o resultado da imagem anterior, Figura 4.6, a primeira coisa lógica a fazer-te é tentar desmontar[3] a partição e fazer a formatação de seguida mas este passo não se revela assim tão simples uma vez que o bloqueio aparece mesmo para o comando *umount*, Figura 4.7, porque um dos processos de sistema continuava a utilizar a partição. Só após leitura da página do manual e muita pesquisa sobre como desmontar é que se descobriu uma solução que permitia cortar a ligação com o diretório mas que continuaria a impedir o *reset* da partição, Figura 4.8.

[3]A formatação, tal como o desmontar/montar, de uma parte do disco só pode ser feita a partir do terminal e a remoção só pode ser feita a partir de ferramentas como o *parted*.

```
ipbrick:/# mkfs.ext3 -c /dev/sda6
mke2fs 1.40-WIP (14-Nov-2006)
/dev/sda6 is mounted; will not make a filesystem here!
ipbrick:/# umount /dev/sda6
umount: /opt: device is busy
umount: /opt: device is busy
ipbrick:/# _
```

Figura 4.7: Bloqueio *umount*.

```
mke2fs 1.40-WIP (14-Nov-2006)
/dev/sda5 is mounted; will not make a filesystem here!
ipbrick:/# umount -l /dev/sda5
ipbrick:/# mount
/dev/sda1 on / type ext3 (rw,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
procbususb on /proc/bus/usb type usbfs (rw)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
/dev/sda3 on /usr type ext3 (rw)
/dev/sda6 on /opt type ext3 (rw)
/dev/sda7 on /home1 type ext3 (rw,usrquota,grpquota,user_xattr,acl)
/dev/sda8 on /home2 type ext3 (rw,usrquota,grpquota,user_xattr,acl)
/home1/_fax on /var/spool/hylafax type none (rw,bind)
nfsd on /proc/fs/nfsd type nfsd (rw)
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
automount(pid4416) on /home type autofs (rw,fd=4,pgrp=4416,minproto=2,maxproto=4
)
ipbrick:/# mkfs.ext3 -c /dev/sda5
mke2fs 1.40-WIP (14-Nov-2006)
/dev/sda5 is apparently in use by the system; will not make a filesystem here!
ipbrick:/# _
```

Figura 4.8: Bloqueio na remoção mesmo após desmontar a partição.

Mais tarde viria a verificar-se que a melhor forma de conseguir a "formatação"consistia em remover a partição por completo o que até funcionava para */var, /opt* mas que gerava problemas para os diretórios */usr, /* pelo simples facto de se ter criado um *feedback* negativo, ou seja, a operação que estava a tentar ser forçada, baseada na capacidade do sistema de ficheiros do *Linux* de permitir a alteração e remoção de ficheiros - Secção 2.1.4 - em simultâneo com a sua utilização por outros processos, estava a provocar que se tentasse apagar o diretório base, /, que contém também o diretório do *chroot* e de todas as ferramentas necessárias para as operações. Para evitar confusão o esquema da Figura 4.9 esclarece o que é descrito neste parágrafo.

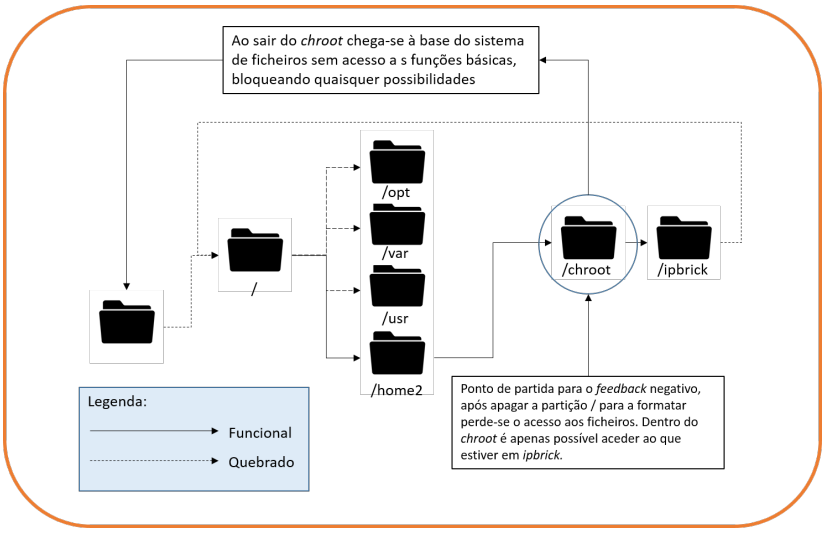

Figura 4.9: *Feedback* negativo.

Esta solução acabou por ser ignorada devido a este *feedback* negativo e daqui surgiu o processo alternativo *SRRD* descrito na secção 3.3.2.2

4.1.1.4 Sincronizar, Reestruturar, Reescrever, Descompactar e Reiniciar (*SRRDR*)

Como o próprio nome da solução indica e conforme o que foi descrito na Secção 3.3.2.2 é necessário fazer testes aos *soft links*, para garantir que o sistema consegue arrancar e usar todas as funções instaladas através destas ligações, é necessário fazer testes à reestruturação dos esquema de partições para alterar o espaço disponível e concentrar isso numa só partição e de seguida reescreve-la de forma a que seja possível descompactar os ficheiros de dados existentes na *iso* para a nova partição. Por último é necessário fazer *reboot* para verificar que o sistema arranca. Todos estes testes são a nível macroscópico, à medida que esta secção for evoluindo vai ser possível perceber-se que os testes feitos foram muitos mais até se conseguir obter a fórmula final. As Figuras 2 e 3 do Anexo 5.2 juntamente com as Figuras 4.4, 4.10 e 4.11 constituem os pontos de partida deste processo.

```
ipbrick:/#
ipbrick:/#
ipbrick:/# mount
/dev/hda1 on / type ext3 (rw,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
procbususb on /proc/bus/usb type usbfs (rw)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
/dev/hda3 on /usr type ext3 (rw)
/dev/hda5 on /var type ext3 (rw)
/dev/hda6 on /opt type ext3 (rw)
/dev/hda7 on /home1 type ext3 (rw,usrquota,grpquota,user_xattr,acl)
/dev/hda8 on /home2 type ext3 (rw,usrquota,grpquota,user_xattr,acl)
/home1/_fax on /var/spool/hylafax type none (rw,bind)
nfsd on /proc/fs/nfsd type nfsd (rw)
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
automount(pid4265) on /home type autofs (rw,fd=4,pgrp=4265,minproto=2,maxproto=4)
ipbrick:/#
ipbrick:/#
ipbrick:/#
```

Figura 4.10: *Mount IPBrick-v5.3*.

```
ipbrick:/# parted
GNU Parted 1.8.8
Using /dev/hda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) p
Model: QEMU HARDDISK (ide)
Disk /dev/hda: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system  Flags
 1      32.3kB  1152MB  1152MB  primary   ext3         boot
 2      1152MB  2221MB  1069MB  primary   linux-swap
 3      2221MB  4943MB  2723MB  primary   ext3
 4      4943MB  21.5GB  16.5GB  extended
 5      4943MB  10.2GB  5289MB  logical   ext3
 6      10.2GB  11.4GB  1152MB  logical   ext3
 7      11.4GB  16.4GB  5026MB  logical   ext3
 8      16.4GB  21.5GB  5059MB  logical   ext3
```

Figura 4.11: Partições *IPBrick-v5.3*.

Em primeiro lugar, foi necessário introduzir o *chroot* por *SSH* tendo em conta a indisponibilidade de ser criado um com ambiente minimalista *wheezy*, primeiro devido à falta do *script* necessário para isso, dentro do *Debian* e em segundo devido à impossibilidade de se obter esse mesmo *script* através dos repositórios normais. foi também criada a pasta *ipbrick* para mais tarde se fazer o *hard link* com a nova partição tal como descrita na Secção 3.3.3.2. O resultado deste processo foi a Figura 4.12.

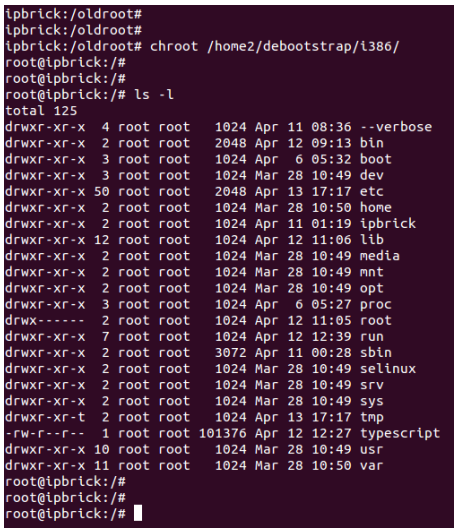

Figura 4.12: Ficheiros do Ambiente Minimalista.

O teste aos *soft links* foram feitos antes de se realizar toda a alteração do esquema de partições e foi feito em duas instâncias. Em primeiro lugar, tal como definido nas etapas de solução, Secção 3.3.3.2, começou por criar-se os diretórios necessários para poder executar o comando *rsync*, com as respetivas opções de forma a preservar todos os *links* fossem *hard* ou *soft*, para dentro da */home2*. Após execução do comando comparou-se os conteúdos de ambas as pastas, a resultante do *rsync* e a original, de forma a garantir que a sincronização tinha sido feita corretamente, esta verificação foi realizada porque existem processos ativos que utilizam as pastas em questão pelo que os ficheiros poderiam não estar todos corretos ou poderia ter havido erros a fazer a sincronização por isto mesmo. Após estabelecer que a operação tinha sido realizada com sucesso tentou criar-se as ligações para os resultados da operação anterior sob o mesmo nome das pastas originais o que resultou numa falha imediata uma vez que o *soft link* e a pasta original não podiam ter a mesma designação. Nesse sentido renomeou-se os diretórios para nomes similares e de seguida voltou a aplicar-se o comando desta vez sem problemas, para garantir que tudo tinha sido preservado reiniciou-se a máquina. Todas as funções e os acesso através dos *soft links* estavam em normal funcionamento no entanto o *link* do *inode* para os diretórios mantinha-se ativo porque não tinha sido alterado, apenas renomeado, como tal era necessário testa-lo caso fosse apagado juntamente com o diretório e com a partição. Entrou-se no ambiente minimalista com recurso ao comando *chroot* e à utilização de um *hard link* de */dev* para o diretório */dev* dentro do *chroot* e passou-se também o ficheiro contido em */etc/fstab* para o respetivo diretório no *chroot* para que se pudesse fazer o próximo passo em segurança e para que mesmo dentro da *jail* fosse possível reconhecer as partições para proceder-se à remoção das partições três, cinco e seis recorrendo ao *parted*. Voltou a reiniciar-se o sistema para verificar se tudo ainda se mantinha em funcionamento. Comprovou-se imediatamente a seguir que tudo estava no funcionamento normal, Figura 4.13 e 4.14.

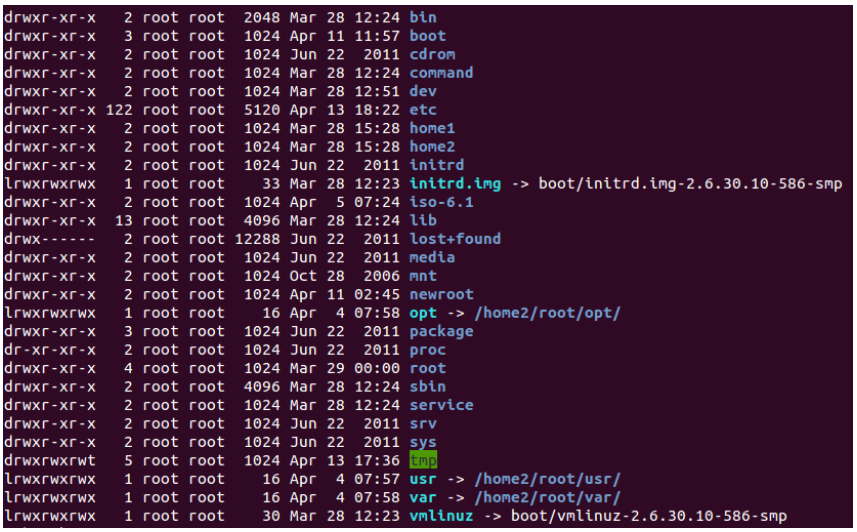

Figura 4.13: *Soft Link* em funcionamento após apagar partições.

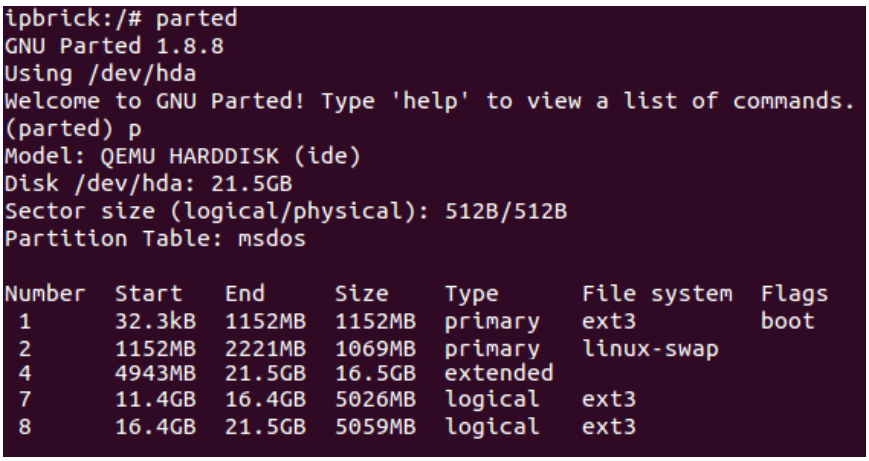

Figura 4.14: Tabela de Partição.

Embora não se tenha descrito anteriormente, tal como na solução anterior *Chroot Bypass*, descrita na Secção 3.3.2.1, a tentativa de apagar partições não é feita sem problemas e mesmo com a utilização do *rsync* o sistema continua a acusar que as partições estão em utilização mas nesta solução não é necessário apagar o diretório base, */*, e o acesso a todos os outros é mantido por isso torna-se mais fácil ultrapassar esta situação como se pode ver pelo exemplo da Figura 4 do Anexo 5.2 na qual é removida a partição cinco ou */var*.

Agora que se conseguiu comprovar que o passo anterior é possível e depois de ter eliminado as partições, redimensionou-se a partição *extended* também com recurso ao *parted* como anteriormente e criou-se uma nova partição primária para depois se descompactar os *drive* e continuar o processo. A criação da nova partição foi feita com os parâmetros:

$$primaryext3(FimparticaoSwap)(InicioparticaoExtendedouhome1) \tag{4.2}$$

A razão pela qual foi utilizado o *EXT3* em vez do *EXT4* como seria possível foi porque tentou-se implementar o *EXT4* e avançar com o processo, quando se chegou ao passo que envolve a instalação e configuração do *GRUB2* obteve-se um impasse, pelo que fez-se a recuperação a partir de um *snapshot* e utilizou-se o *EXT3* para testar a possibilidade de o motivo pelo qual o *bootloader* não reconhecia o sistema seria pelo facto de o disco ficar com duas versões diferentes do *EXT*, como tal todos os resultados são apresentados com *EXT3*. Embora se tenha concluído que a utilização da última versão do sistema de ficheiros nativo nada teve haver com o problema não se fez a atualização com esta devido à escassez de tempo que já tinha sido causada pelos diversos problemas descritos nos parágrafos seguintes.

Após a criação extraiu-se os ficheiros *drive* (*.dat*) do um ao quatro do ficheiro *IPBrick-v6.1* que contém não só estes mas também os restantes para a criação das partições de dados bem como alguns ficheiros de instalação e um ficheiro *MD5SUM*. Verificou-se a integridade dos dados para ter a certeza que não tinham sido corrompidos e também como forma de preparar já o código para o *script* de implementação. Posto isto fez-se a extração dos ficheiros que dariam origem as novas pastas */, /usr, /var, /opt*. Este último passo foi efetuado com recurso a uma ferramenta representada pelo comando *tar*[4] este comando é responsável pela extração de ficheiros como os indicados na sua forma predefinida não mantém as permissões com que os ficheiros foram compactados o que gerou um problema de ligação à base de dados quando se conseguiu provar o funcionamento da solução, quando se descobriu o problema em questão foi necessário reverter para o *snapshot* tirado na altura e refazer todo o processo até comprovar novamente a solução mas desta vez com recurso à opção que garante as permissões corretas *–numeric-owner*.

Resta ainda instalar o *GRUB* e testar o reconhecimento do *OS* e respetivo arranque. na instalação do *GRUB* o *chroot* torna-se essencial porque sem este não seria possível introduzir a última versão do *bootloader* no sistema uma vez que este não aparece disponível para esta versão do *Linux*. Utilizando o comando para a instalação sem parâmetros adicionais implica que o *GRUB* seja instalado no *chroot* em vez de o ser na nova partição e que ao reiniciar, que foi o teste executado para verificar se era possível iniciar, o *bootloader* não encontre ficheiros nenhuns disponíveis, para além disso não se tinha obtido a configuração correta.

[4]

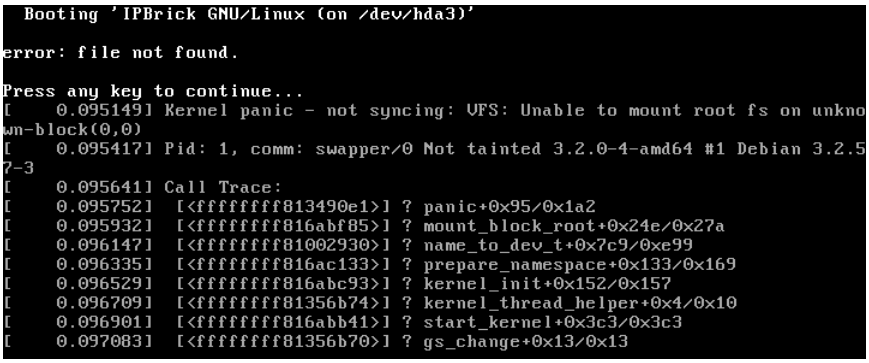

Figura 4.15: *GRUB2* não encontra o ficheiro, *lock* no *boot*.

Estes erros, Figura 4.15, foram definitivamente os que deram mais trabalho e dores de cabeça, o *grub-mkconfig* não reconhecia o sistema operativo descompactado para a nova partição o que resultava em que sempre que se reinicia-se o sistema resultaria ou na falta de ficheiros e em ficar numa prisão entre o início do sistema operativo e o *grub*. As Figuras 4.16 e 4.17 mostram a respetiva instalação e configuração correta do *GRUB2* após todas as alterações feitas.

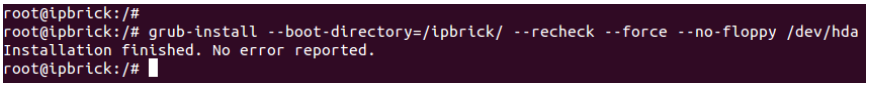

Figura 4.16: Instalação do *GRUB2* no diretório pretendido dentro do *chroot*.

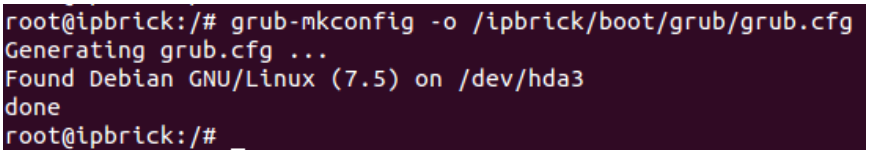

Figura 4.17: Obtenção do ficheiro de configuração após aplicação de *grub-mkconfig*.

Mais tarde, quando foi possível criar as configurações corretas mesmo assim não se conseguia iniciar a *IPBrick-v6.1* o que viria a verificar-se que era a falta da execução do comando *update-initramfs* que permite criar o ficheiro *initrd*, ou melhor o disco de *RAM* inicial, Figura 4.18. Na execução deste comando teve apenas de se ter o cuidado de indicar o diretório correto porque caso contrário o *initrd* é criado mas sem todos os módulos necessários e o sistema não inicia na mesma porque não sabe onde se encontra o ficheiro especial que descreve a partição pretendida.

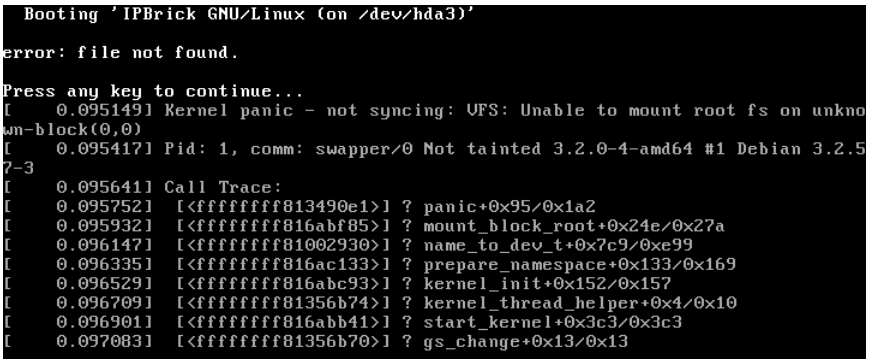

Figura 4.18: Problema resultante da desconfiguração do *initramfs*. Falta de módulos causa o *crash*.

Já na reta final, não podiam deixar de surgir problemas na execução de testes e no desenvolvimento do algoritmo de *update*. Teoricamente pelo que foi definido e explicado no Capítulo 3.1.2, tendo em conta as etapas realizadas até este ponto, bastava apenas editar corretamente o ficheiro *fstab* de forma a que se provasse que a solução era possível e começar a trabalhar no *script*. Acontece que o mesmo erro que ocorreu após correção do *initrd* mantinha-se, erro esse que se atribuiu no momento à falta de o ficheiro *fstab* não estar correto. Depois de muita pesquisa sobre o *Linux Wheezy*, sobre o *GRUB2* e sobre como estes funcionam, que ficheiros apoiam a sua configuração e a deteção de partições e ficheiros percebeu-se que em primeiro lugar neste sistema operativo o ficheiro *mtab* é automaticamente transformado num *soft link* para */proc/mount/* por questões de coerência e para que o sistema se mantenha sempre sincronizado e isto é provocado pelo ficheiro */etc/init.d/checkroot.sh*. As Figuras 4.19 e 4.20 do Anexo 5.2 permitem ver os erros referenciados anteriormente.

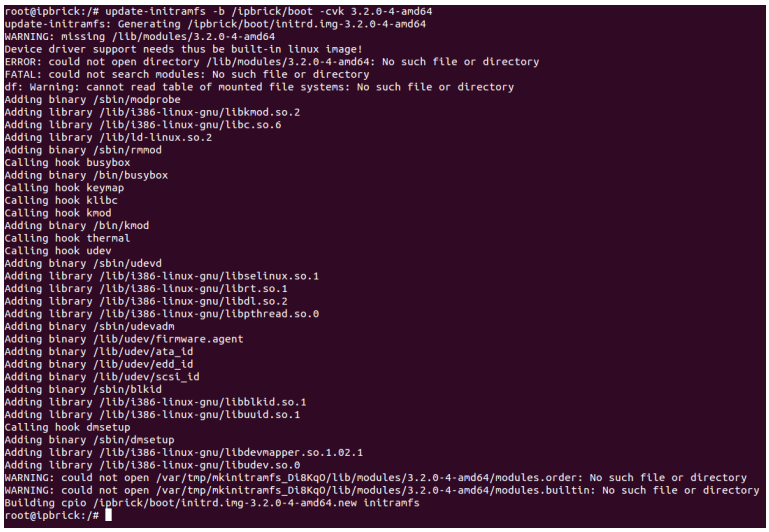

Figura 4.19: Erro do comando *update-initramfs*.

```
### BEGIN /etc/grub.d/30_os-prober ###
menuentry "Debian GNU/Linux (7.5) (on /dev/hda3)" --class gnu-linux --class gnu --class os {
        insmod part_msdos
        insmod ext2
        set root='(hd0,msdos3)'
        search --no-floppy --fs-uuid --set=root 8075f471-a739-4d5f-bb2a-fc9acaf033c4
        linux /boot/vmlinuz-3.2.0-4-amd64 root=/dev/hda3
        initrd /boot/initrd.img-3.2.0-4-amd64
}
### END /etc/grub.d/30_os-prober ###

### BEGIN /etc/grub.d/40_custom ###
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.
### END /etc/grub.d/40_custom ###

### BEGIN /etc/grub.d/41_custom ###
if [ -f  $prefix/custom.cfg ]; then
  source $prefix/custom.cfg;
fi
### END /etc/grub.d/41_custom ###
root@ipbrick:/#
```

Figura 4.20: Ficheiro de configuração do *GRUB* com o *menu entry* para onde aponta que o *root=/dev/hda3*.

Relativamente ao *bootloader* quando o *grub.cfg* é gerado, segundo este processo, a nomenclatura dos discos era baseada em *IDE* e resultava em ficheiros especiais *hda* enquanto que após o *update* o *bootloader* estava à procura de um ficheiro do tipo *sda* para conseguir encontrar o diretório de raiz. Arranjou-se solução para estes problemas no caso do *checkroot.sh* comentando a linha que corresponde à instrução *mtab_migrate* o que impede a criação do *soft link*, após isso reutilizou-se o comando *update-initramfs* para atualizar a imagem de *initrd*. Por fim, depois de se ter alterado o ficheiro *grub.cfg* em todas as linhas que continham */hda3* para */sda3*, reiniciou-se a máquina e finalmente se obteve sistema operativo na sua última versão, entrando diretamente para a interface gráfica como é suposto pela primeira vez para se conseguir configurar o sistema operativo.

As Figuras 4.21 e 4.22 permitem ver o resultado final nas partições e nos ficheiros. Para ver informação mais detalhada pode ainda verificar-sa as Figuras 7, 6 e 5 do Anexo 5.2.

```
ipbrick:/# dpkg -l | grep system-manager
ii  system-manager                                  6.1
     all          System Manager v6.1
ipbrick:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs           12G  6.5G  5.0G  57% /
udev             10M     0   10M   0% /dev
tmpfs           101M  1.9M   99M   2% /run
/dev/sda3        12G  6.5G  5.0G  57% /
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           507M     0  507M   0% /run/shm
/dev/sda5        18G  3.5G   13G  21% /home1
/dev/sda6        18G  2.1G   15G  13% /home2
/dev/sda5        18G  3.5G   13G  21% /var/spool/hylafax
ipbrick:/# blkid
/dev/sda1: UUID="2288ea52-6975-4a2b-9ba5-eab4bac0fddf" SEC_TYPE="ext2" TYPE="ext
3"
/dev/sda2: UUID="ce5617ed-a3c8-48b2-92a7-6d977bcbb1a1" TYPE="swap"
/dev/sda3: UUID="8075f471-a739-4d5f-bb2a-fc9acaf033c4" TYPE="ext3"
/dev/sda5: UUID="77c190a8-168e-4ba5-9a1d-2593cf3920c7" SEC_TYPE="ext2" TYPE="ext
3"
/dev/sda6: UUID="1baa4e8a-7404-43f9-aeff-ccdd7190e37e" TYPE="ext3" SEC_TYPE="ext
2"
/dev/sr0: LABEL="IPBRICK" TYPE="iso9660"
ipbrick:/#
```

Figura 4.21: Resultado da atualização para o *IPBrick-v6.1*.

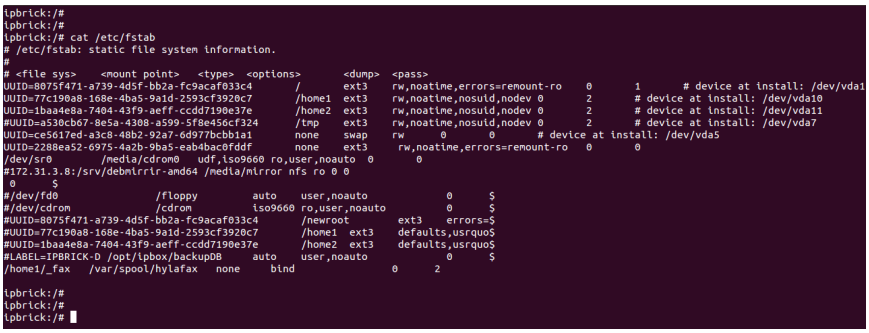

Figura 4.22: Ficheiro responsável pela montagem de partições no *boot*.

4.2 Implementação do MGA*SI*

A implementação do Módulo de Gestão de Atualizações implica que sejam lançadas duas *ISO* diferentes, uma para versões 5 do sistema operativo e outra para as versões 6. Para a versão 5 é necessário que a *ISO* tenha os *drive* correspondentes ao próximo *update*, *IPBrick-v6.3* por exemplo, bem como os ficheiros para *chroot* e o respetivo *script* para fazer *update*. Para versões superiores à *IPBrick-v6.0* é necessário, ao fazer a atualização, reestruturar o sistema de partições com recurso a um *script* e instalar o módulo de gestão que consiste na interface gráfica, no *script* para atualização para novas distribuições e o algoritmo de interação com o repositório e de introdução dos *minor updates*.

Até ao momento o que ficou implementado, no que toca a *major updates* foi o *script* de instalação que permite ver o espaço livre existente através dos comandos já existentes no *linux* e todo o restante processo até ser necessário reiniciar o sistema, ou seja, remoção das partições */usr, /var, /opt*, reestruturação do espaço nas partições de dados e concentração do espaço todo numa nova partição criando-a como *EXT3*, extração dos ficheiros de todas as *drives*, a instalação do *grub2*, alteração dos ficheiros *fstab, grub.cfg, checkroot.sh* para que o sistema inicie a partir da nova partição. Ficou também definido quais são os elementos necessários para criar o ficheiro *ISO* que vai ser usado pelo cliente para conseguir fazer o *update* e obter a versão mais recente.

No que toca à interface gráfica foi feito o protótipo e todo o seu *design*, bem como as normas para o funcionamento da mesma, que tipo de opções irá permitir e de que forma irá interagir com o Módulo de Gestão de Atualizações.

O MGA*SI* como um todo encontra-se ainda em desenvolvimento tendo em conta aquilo que o constitui e que será necessário, existe neste momento parte do código *PHP* que será necessário para ir buscar as informações à base de dados do lado do cliente, dentro do mesmo protótipo está também a ser desenvolvido o algoritmo de comunicação com o repositório, ou seja, ligação ao servidor, envio de informação e *download* das atualizações com verificação do *MD5SUM*. Está também a ser desenvolvida a instalação dos *major/minor updates* mediante as configurações guardadas num ficheiro que é carregado para o módulo sempre que este se encontra ativo. Do lado do servidor também se encontra definida a maneira de operar, mapear a informação e enviar

os ficheiros de atualização para o cliente isto com recurso a um ficheiro *XML* para aumentar a eficiência do sistema.

4.3 Conclusão

Dos testes realizados na primeira Secção resultaram as seguintes considerações:

1. *IPBrick-v5.3 -> IPBrick-v6.1* - Reconhece a existência de uma versão anterior e não altera as partições de dados.
2. *IPBrick-v6.1 -> IPBrick-v5.3* - Não reconhece a existência de uma versão superior e reescreve todo o disco apagando os dados.
3. *IPBrick-v6.1 -> IPBrick-v6.1 + updateXY* - O utilizador faz o *download* e depois insere-o no painel de *update*, Figura 2.32, que rejeita caso os anteriores não tenham sido inseridos, aceita se estes foram ou rejeita caso já exista uma versão introduzida mais recente.
4. As atualizações das aplicações e do sistema (*minor*) levam bastante tempo a serem introduzidas, primeiro porque têm de o ser uma a uma e segundo porque a sua instalação pode levar até 10 minutos.
5. *IPBrick-v5.3 -> IPBrick-v5.5* - Implica a instalação de cerca de 20 *minor updates* o que corresponde a cerca de 3 horas de atualizações partindo do pressuposto que o utilizador as põe todas seguidas.
6. No mínimo cerca de 15 *GB* de espaço livre em disco serão necessários.

O uso do ambiente minimalista *Wheezy* é indispensável para conseguir criar um sistema de ficheiros *EXT3/EXT4* devido ao facto da versão 5.3 da *IPBrick* não possuir os ficheiros necessários para criar sistemas deste tipo, apenas consegue criar *EXT2*. Para além disso é também um recurso valioso porque constitui a ponte para conseguir instalar o *GRUB 2.0* no disco. Só com a instalação deste *software* é que o disco será capaz de reconhecer e arrancar o novo *OS*, ou melhor, a sua versão mais atualizada. O *chroot* será particularmente crucial em todas as operações que envolvem o *boot* uma vez que na *IPBrick-v5.3 (Linux Debian Etch)* não existe o *GRUB2* apenas o *GRUB Legacy*, ver Secção 2.1.2.2, ou os comandos (*update-grub2, update-initramfs*) necessários para a correta configuração. Também a utilização da sincronização das pastas */usr, /var, /opt* é extremamente importante porque garante que as funções necessárias para o processo se encontram disponíveis para finalizar a atualização.

Desta forma garante-se que os dados do utilizador e as suas configurações guardadas em *homeY* não são modificadas nem apagadas, que não é gerado qualquer tipo de custo a mais, todos os utilizadores são abrangidos, a atualização é relativamente *user-friendly* uma vez que implica apenas a execução de um *script* e descarregamento do *chroot*, consegue-se ultrapassar as limitações de dependência do *Linux* e nenhuma parte da operação depende da Empresa a não ser o desenvolvimento do módulo e garantia do seu funcionamento.

A reestruturação do esquema de partições e a criação de uma partição escondida consegue dar maior dinamismo ao sistema para futuras atualizações e implementação de outros serviços e funcionalidades.

Finalizado tudo o que foi descrito anteriormente e conseguida a resolução de todos os problemas que ocorreram e dificultaram este desenvolvimento, embora a descrição anterior não faça justiça ao trabalho necessário e ao tempo perdido em pesquisa e correção de erros, foi possível criar um *bash script* com todo o processo de instalação para os *major updates* que após executado será necessário um *reboot*. Juntamente com parte do código geral que constituirá o MGA*SI* bem como todo o design associado à interface gráfica e todas as regras necessárias ao funcionamento do módulo.

Capítulo 5

Conclusões

5.1 Satisfação dos Objetivos

Os objetivos definidos no Capítulo 1 dividem-se em três grandes blocos: pesquisa, testes e investigação e implementação.

No que toca à pesquisa das tecnologias existentes e como estas funcionam os objetivos foram completamente cumpridos como é possível ver pela informação contida no Capítulo 2 onde é descrito funcionalidades, características, ferramentas e notas importante a ter sobre o *Linux*, tipos de virtualização e funcionamento dos mesmos e inclusive mecanismos que podem ser utilizados para o desenvolvimento de formas de garantir que os serviços nunca são parados do ponto de vista do cliente aquando de uma atualização. Ainda neste Capítulo fez-se uma análise sobre linguagens de programação que poderiam ser utilizadas e também sobre os produtos *IPBrick* disponíveis bem como módulos para atualização de *software* de outras empresas associadas ao mercado comercial e empresarial sob diferentes contextos.

Relativamente aos testes a serem feitos e investigação, Capítulo 3 e 4, pretendia-se inicialmente explorar formas onde seria possível fazer a atualização sem ter de se parar serviços mas as soluções encontradas implicavam um aumento de custos ou interação com ferramentas às quais não era suposto ou não era permitido e portanto entravam diretamente em confronto com as limitações. Numa instância posterior, criou-se de uma ideia onde fosse possível fazer atualizações evitando a dependência do *Linux* e sem aumentar o espaço físico disponível, solução esta que foi testada, comprovada e de onde advém uma das componentes do Módulo para Gestão de Atualizações de *Software IPBrick* e que pode ser aplicada também noutros sistemas que seja baseados em *Linux*. Relativamente à idealização da interface gráfica, do algoritmo macroscópico do módulo bem como opções para tornar os *updates* mais rápidos de serem introduzidos no sistema todos os objetivos foram cumpridos podendo verificar-se isso no Capítulo 3 através das explicações dadas para cada um e pelos diagramas de funcionamento.

Relativamente à implementação, Capítulo 4, tal como descrito nas conclusões do mesmo, o desenvolvimento referente às atualizações que pretendiam ser provadas está definido. O código para o funcionamento do módulo (ligação cliente-repositório) e para poder ser introduzido no

sistema ainda se encontra em desenvolvimento bem como a interface gráfica, de uma forma mais objetiva pode se dizer que a estrutura está definida bem como os elementos de ligação mas estes elementos não estão totalmente desenvolvidos nem interagem entre si portanto, de uma forma mais objetiva, poderá pôr-se a classificação na implementação da interface gráfica de 3 em 10 e no desenvolvimento para o funcionamento do código em geral 5 em 10.

5.2 Trabalho Futuro

Nem todos os objetivos foram completamente finalizados portanto o projeto a que se refere esta dissertação será ainda finalizada pelo estudante mediante os objetivos mencionados no Capítulo 1. No entanto, como é normal no mundo tecnológico e nos outros, não existem versões finais ou objetivos suficientes e existe sempre espaço para fazer mais e melhor. Isto pode ser traduzido que num futuro próximo na implementação algumas melhorias ou pelo menos *features* que do ponto de vista do cliente serão bastante atrativas no que toca ao Módulo para Gestão de Atualizações de *Software IPBrick*, nomeadamente:

- *Disaster Recovery* - Comunica com o módulo de atualização e aquando de uma operação destas o módulo automaticamente faz *download* dos *updates* que estavam instalados anteriormente caso não os tenha em disco e instala-os.
- Aplicar derivados do algoritmo de *Live Migration* ao nível das máquinas existentes na *Cloud* de forma a que seja possível fazer atualizações sem paragem de serviços ou sem ser necessário reiniciar a máquina mesmo quando se tratam de atualizações onde a distribuição é alterada.

Relativamente ao *IPBrick OS* foram já aplicadas algumas alterações ao esquema de partições para as *major updates* fossem possíveis, o que abra uma janela onde se passe a aplicar um novo esquema para todas as versões do sistema operativo deste ponto para o futuro: uma partição escondida, *swap*, sistema e dados ou *home*. A razão pela a escolha deste esquema é simples reduz a complexidade dos processos de instalação, manutenção e acrescenta a possibilidade de ter um espaço em disco reservado e invisível para o utilizador que poderá ter como objetivo permitir que haja sempre fundo de maneio em termos de espaço livre para atualizações e outras aplicações.

Anexo A - Tabelas

Package Management Systems		
Sintaxe *aptitude*	Sintaxe *apt-get/apt-cache*	Significado
aptitude update	apt-get update	update package archive metadata
aptitude install foo	apt-get install foo	install candidate version of "foo"package with its dependencies
aptitude safe-upgrade	apt-get upgrade	install candidate version of installed packages without removing any other packages
aptitude full-upgrade	apt-get dist-upgrade	install candidate version of installed packages while removing other packages if needed
aptitude remove foo	apt-get remove foo	remove "foo"package while leaving its configuration files
N/A	apt-get autoremove	remove auto-installed packages which are no longer required
aptitude purge foo	apt-get purge foo	purge "foo"package with its configuration files
aptitude clean	apt-get clean	clear out the local repository of retrieved package files completely
aptitude autoclean	apt-get autoclean	clear out the local repository of retrieved package files for outdated packages
aptitude show foo	apt-cache show foo	display detailed information about "foo"package
aptitude search <regex>	apt-cache search <regex>	search packages which match <regex>
aptitude why <regex>	N/A	explain the reason why <regex> matching packages should be installed
aptitude why-not <regex>	N/A	explain the reason why <regex> matching packages can not be installed
aptitude search 'i!M'	apt-mark showmanual	list manually installed packages

Tabela 1: Comandos *APT* e *aptitude*[10].

Anexo B - Figuras

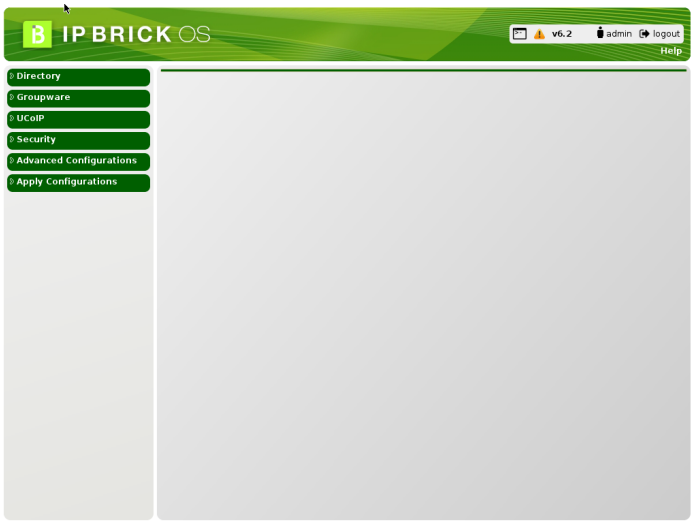

Figura 1: *IPBrick v6.2*.

```
ipbrick:/#
ipbrick:/#
ipbrick:/# ls -l
total 43
drwxr-xr-x   2 root root  2048 2016-06-15 15:54 bin
drwxr-xr-x   3 root root  1024 2016-06-15 16:49 boot
drwxr-xr-x   2 root root  1024 2011-06-22 17:46 cdrom
drwxr-xr-x   2 root root  1024 2016-06-15 15:54 command
drwxr-xr-x  14 root root 13660 2016-06-15 18:18 dev
drwxr-xr-x 122 root root  5120 2016-06-23 16:58 etc
drwxr-xr-x   2 root root     0 2016-06-15 18:18 home
drwxr-xr-x  21 root root  1024 2011-06-22 17:47 home1
drwxr-xr-x   8 root root  1024 2011-06-22 17:46 home2
drwxr-xr-x   2 root root  1024 2011-06-22 15:49 initrd
lrwxrwxrwx   1 root root    33 2016-06-15 15:53 initrd.img -> boot/initrd.img-2.6.30.10-586-smp
drwxr-xr-x  13 root root  4096 2016-06-15 15:54 lib
drwx------   2 root root 12288 2011-06-22 15:02 lost+found
drwxr-xr-x   2 root root  1024 2011-06-22 15:49 media
drwxr-xr-x   2 root root  1024 2006-10-28 15:06 mnt
drwxr-xr-x  16 root root  1024 2011-06-22 18:51 opt
drwxr-xr-x   3 root root  1024 2011-06-22 16:01 package
dr-xr-xr-x 169 root root     0 2016-06-15 18:17 proc
drwxr-xr-x   3 root root  1024 2011-06-22 18:51 root
drwxr-xr-x   2 root root  4096 2016-06-15 15:54 sbin
drwxr-xr-x   2 root root  1024 2016-06-15 15:54 service
drwxr-xr-x   2 root root  1024 2011-06-22 15:49 srv
drwxr-xr-x  12 root root     0 2016-06-15 18:17 sys
drwxrwxrwt   6 root root  1024 2016-06-15 18:36 tmp
drwxr-xr-x  13 root root  1024 2011-06-22 15:52 usr
drwxr-xr-x  19 root root  1024 2011-06-22 17:31 var
lrwxrwxrwx   1 root root    30 2016-06-15 15:54 vmlinuz -> boot/vmlinuz-2.6.30.10-586-smp
ipbrick:/#
ipbrick:/#
ipbrick:/#
```

Figura 2: Ficheiros *IPBrick-v5.3*.

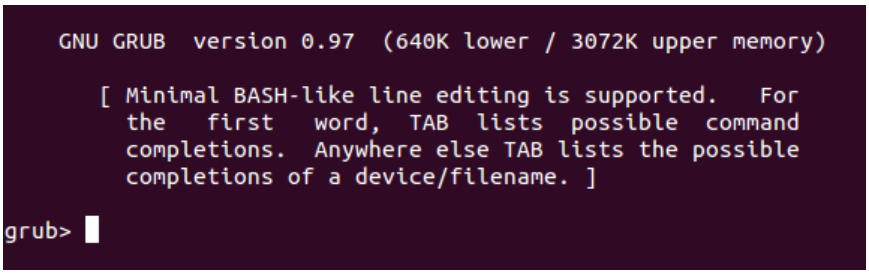

Figura 3: *Grub Legacy 0.97*.

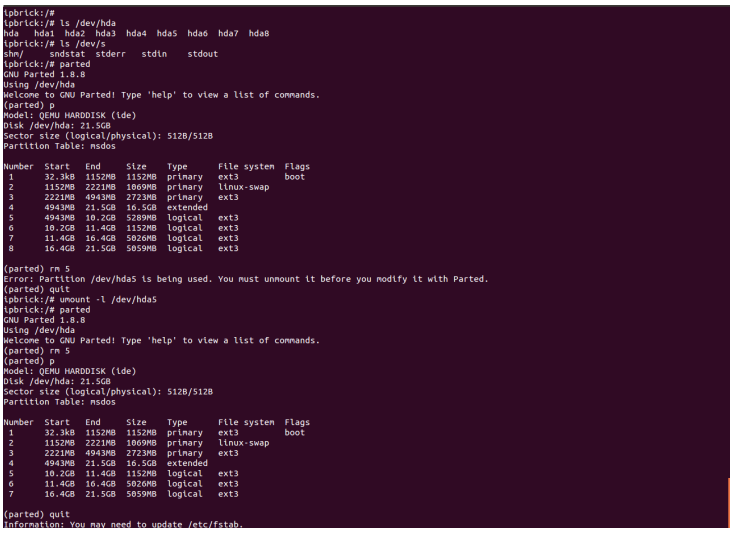

Figura 4: Remoção de uma partição sem problemas.

Figura 5: *Mount* resultante da atualização para o *IPBrick-v6.1*.

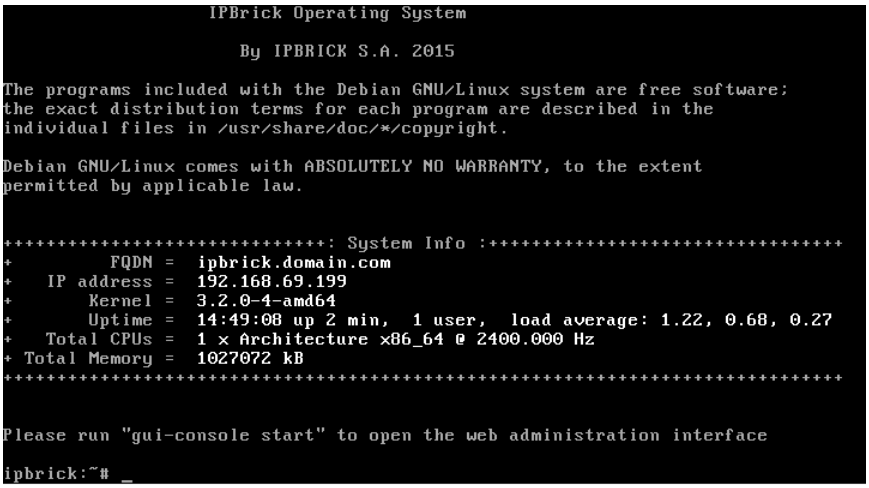

Figura 6: Terminal após *login* na *IPBrick-v6.1*.

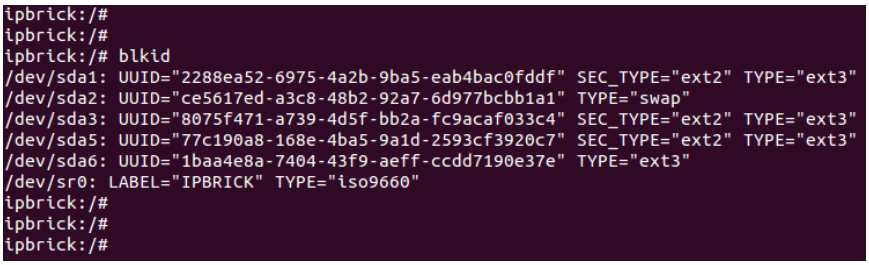

Figura 7: *UUID* das partições na *IPBrick-v6.1*.

Referências

[1] Inc Sun Microsystems. *Solaris handbook for Sun peripherals*. Sun Microsystems ; Part No. 806-2219-10, February 2000, Revision A. Sun Microsystems, Incorporated, 2000. Disponível em: 26-06-2016. URL: https://docs.oracle.com/cd/E19455-01/806-2219-10/806-2219-10.pdf.

[2] Ramey (Reserve University) Chet, Western (Reserve University) Case, e Fox (Free Software Foundation) Brian. *Bourne-Again SHell manual*. Free Software Foundation, 4.3 edição, 2014. Disponível em: 26-06-2016. URL: https://www.gnu.org/software/bash/manual/bash.pdf.

[3] VMware. Understanding full virtualization, paravirtualization, and hardware assis, 2007. Disponível em: 26-06-2016. URL: https://www.vmware.com/files/pdf/VMware_paravirtualization.pdf.

[4] Diego Perez-Botero. A brief tutorial on live virtual machine migration from a security perspective. *University of Princeton, USA*, página 8, 2011. Disponível em: 20-02-2016. URL: http://www.cs.princeton.edu/~diegop/data/580_midterm_project.pdf.

[5] Mohan Anju e S. Shine. Survey on live vm migration techniques. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 2(1):155–157, 2013. doi:http://ijarcet.org/wp-content/uploads/IJARCET-VOL-2-ISSUE-1-155-157.pdf.

[6] Seagate. Serial ata, 2012. Disponível em: 26-06-2016. URL: https://web.archive.org/web/20120105073432/http://www.seagate.com/content/pdf/whitepaper/SerialATA_comparison_UATA_Technology.pdf.

[7] Coomes J., Chalupa C., Coomes J., e Houlder G. *Serial Attached SCSI (SAS) Interface Manual*. Seagate Technology LLC, May 2006. Disponível em: 26-06-2016. URL: http://www.seagate.com/staticfiles/support/disc/manuals/sas/100293071b.pdf.

[8] Benson Calum, Clark Bryan, e Nickell Seth. Gnome quick swot analysis, april 2010, 2010. URL: https://wiki.gnome.org/Engagement/SWOT.

[9] Davis Ziff. Mobile super guide. *PC Mag*, 26(18):96, 2007. Disponível em: 26-06-2016. URL: https://books.google.pt/books?id=-6zvRFEfQ24C.

[10] Free Software Foundation. Chapter 2. debian package management, 2013. Disponível em: 26-06-2016. URL: https://www.debian.org/releases/wheezy/i386/release-notes/ch-upgrading.html.

[11] J.R. Olson, T.M. Naftel, D. Teater, S.D. Nguyen, e J.P. Ohr. Proxy backup of virtual disk image files on nas devices, Maio 7 2013. Disponível em: 26-06-2016. URL: https://docs.google.com/viewer?url=patentimages.storage.googleapis.com/pdfs/US8438349.pdf.

[12] R. Radhakrishnan, R. Pepper, e A. Rajan. Interface for virtual machine administration in virtual desktop infrastructure, Outubro 28 2010. Disponível em: 26-06-2016. URL: https://docs.google.com/viewer?url=patentimages.storage.googleapis.com/pdfs/US20100275200.pdf.

[13] VirtualBox. Chapter 5. virtual storage, 2004–2016. URL: https://www.virtualbox.org/manual/ch05.html.

[14] P. CHAWLA e J. CHAMCHAM. Accessing virtual disk content of a virtual machine without running a virtual desktop, Julho 28 2011. Disponível em: 26-06-2016. URL: https://www.google.com/patents/US20110185355.

[15] A.E. Dittmer. Method for dynamically generating a configuration for a virtual machine with a virtual hard disk in an external storage device, February 2013. Disponível em: 26-06-2016. URL: https://docs.google.com/viewer?url=patentimages.storage.googleapis.com/pdfs/US8370835.pdf.

[16] Seagate. Serial ata, January 2003. Disponível em: 26-06-2016. URL: http://www.ece.umd.edu/courses/enee759h.S2003/references/serialata10a.pdf.

[17] G.E. Clarke e E. Tetz. *CompTIA A+ Certification All-In-One For Dummies*. –For dummies. Wiley, 2nd edição, 2009. Disponível em: 26-06-2016. URL: https://books.google.com/books?id=NmFTnnfmmmYC.

[18] Javvin Technologies. *Network Protocols Handbook*. Javvin Technologies, 2nd edição, 2005. Disponível em: 26-06-2016. URL: http://bkarak.wizhut.com/www/lectures/networks-07/NetworkProtocolsHandbook.pdf.

[19] J. Hoskins. *Exploring IBM Server & Storage Technology: A Layman's Guide to the IBM EServer and TotalStorage Families*. Exploring IBM Series. Maximum Press, 2005. Disponível em: 26-06-2016. URL: https://books.google.de/books?id=9affpwNAloAC.

[20] Remzi H. Arpaci-Dusseau e Andrea C. Arpaci-Dusseau. *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books, 0.91 edição, May 2015. Disponível em: 26-06-2016. URL: http://pages.cs.wisc.edu/~remzi/OSTEP/.

[21] A. Rusling David. *The Linux Kernel*. David A. Rusling, 1999. Disponível em: 26-06-2016. URL: http://www.tldp.org/LDP/tlk/tlk.html.

[22] William E. Shotts Jr. *The Linux Command Line*. No Starch Press, 2012.

[23] Schmidt Klaus. *High Availability and Disaster Recovery: Concepts, Design, Implementation*. Debian, 2006. Disponível em: 26-06-2016. URL: https://wiki.debian.org/chroot, doi:10.1007/3-540-34582-5.

[24] J. Lasser. *Think UNIX*. Que-Consumer-Other Series. Que, 2000. Disponível em: 26-06-2016. URL: https://books.google.de/books?id=_ZF2CJ0XfGoC.

[25] Kügler Sebastian e Christensen Claus. The plasma handbook, 2010. URL: https://docs.kde.org/stable5/en/kde-workspace/plasma-desktop/plasma-desktop.pdf.

[26] C. Negus. *Linux Bible*. Bible. John Wiley & Sons, 2015. Disponível em: 26-06-2016. URL: https://books.google.pt/books?id=2BsFCAAAQBAJ.

[27] R. Grant e P. Bull. *Ubuntu Made Easy: A Project-based Introduction to Linux*. No Starch Press Series. No Starch Press, 2012. Disponível em: 26-06-2016. URL: https://books.google.pt/books?id=NGpbBAAAQBAJ.

[28] Z.L. Berge e L. Muilenburg. *Handbook of Mobile Learning*. Taylor & Francis, 2013. Disponível em: 26-06-2016. URL: https://books.google.pt/books?id=mW4XnKr9Hi0C.

[29] Benson Calum, Clark Bryan, e Nickell Seth. Gnome human interface guidelines 2.2.3, 2014. URL: https://developer.gnome.org/hig-book/3.12/hig-book.html.

[30] International Labour Office e International Ergnonomics Association. *Ergonomic Checkpoints*. ILO - International Labour Office, 2nd edição, July 2010. Disponível em: 26-06-2016. URL: http://www.ilo.org/wcmsp5/groups/public/---dgreports/---dcomm/---publ/documents/publication/wcms_120133.pdf.

[31] John D. Lee. Emerging challenges in cognitive ergonomics: Managing swarms of self-organizing agent-based automation. *Theoretical Issues in Ergonomics Science*, 2(3):238–250, 2001. Disponível em: 26-06-2016. URL: http://dx.doi.org/10.1080/14639220110104925, arXiv:http://dx.doi.org/10.1080/14639220110104925, doi:10.1080/14639220110104925.

[32] Charles M. Kozierok. New technology file system (ntfs), 2001. Disponível em: 26-06-2016. URL: http://www.pcguide.com/ref/hdd/file/ntfs/index.htm.

[33] H.T. Sencar e N. Memon. *Digital Image Forensics: There is More to a Picture than Meets the Eye*. SpringerLink : Bücher. Springer New York, 2012. Disponível em: 26-06-2016. URL: https://books.google.de/books?id=PzP9ViF8oAIC.

[34] Linuxfilesystemsexplained, 2015. Disponível em: 26-06-2016. URL: https://help.ubuntu.com/community/LinuxFilesystemsExplained.

[35] Phillips Daniel. A directory index for ext2, 2001. Disponível em: 26-06-2016. URL: https://www.usenix.org/legacy/publications/library/proceedings/als01/full_papers/phillips/phillips.pdf.

[36] Tim Jones M. Anatomy of linux journaling file systems, june 2008. Disponível em: 26-06-2016. URL: http://www.ibm.com/developerworks/library/l-journaling-filesystems/l-journaling-filesystems-pdf.pdf.

[37] Sollins K. *THE TFTP PROTOCOL (REVISION 2)*. The Internet Society, 1992. Disponível em: 26-06-2016. URL: https://tools.ietf.org/pdf/rfc1350.pdf.

[38] Droms R. *Dynamic Host Configuration Protocol*. The Internet Society, 1997. Disponível em: 26-06-2016. URL: https://tools.ietf.org/pdf/rfc2131.pdf.

[39] Croft Bill e Gilmore John. *BOOTSTRAP PROTOCOL (BOOTP)*. The Internet Society, 1985. Disponível em: 26-06-2016. URL: https://tools.ietf.org/pdf/rfc951.pdf.

[40] Finlayson Ross, Mann Timothy, Mogul Jeffrey, e Theimer Marvin. *A Reverse Address Resolution Protocol*. The Internet Society, 1984. Disponível em: 26-06-2016. URL: https://tools.ietf.org/pdf/rfc1350.pdf.

[41] Luis Martins. Ipbrick cloud environment, 2010. Disponível em: 26-06-2016. URL: https://paginas.fe.up.pt/~ee04114/repositorio/Dissertacao_v_provi.pdf.

[42] Rukunuddin Muhammad, P. Ghalib, e Karan Thakkar Swarnalatha. Virtual machine migration with an open source hypervisor. *International Journal of Engineering and Technology (IJET)*, 5(3):2671–2677, 2013. doi:http://www.enggjournals.com/ijet/docs/IJET13-05-03-209.pdff.

[43] Hess Ken. Understanding hardware-assisted virtualization. Disponível em: 26-06-2016. URL: http://www.admin-magazine.com/Articles/Hardware-assisted-Virtualization.

[44] Madan Deepanshu, Pant Ashish, Kumar Suneet, e Arora Arjun. E-learning based on cloud computing. *IJARCSSE - International Journal of Advanced Research in Computer Science and Software Engineering*, 2(2):6, 2012. Disponível em: 26-06-2016. URL: http://www.chinacloud.cn/upload/2012-03/12030611581796.pdf.

[45] IPBRICK S.A. *IPBRICK - Reference Guide Version 6.0*. IPBRICK S.A., 2014. Disponível em: 26-06-2016. URL: http://downloads.ipbrick.com/IPBrick/documentation/EN/manual_referencia_6_EN.pdf.

[46] S.A. IPBrick. About ipbrick. Disponível em: 26-06-2016. URL: http://www.ipbrick.com/about-ipbrick/.

[47] S.A. IPBrick. See what ipbrick can do for you and your company. Disponível em: 26-06-2016. URL: http://www.ipbrick.com/products/.

[48] S.A. IPBrick. Ipbrick.i. Disponível em: 26-06-2016. URL: http://www.ipbrick.com/products/ipbrick-i/.

[49] S.A. IPBrick. Ipbrick.gt. Disponível em: 26-06-2016. URL: http://www.ipbrick.com/ipbrick-gt/.

[50] S.A. IPBrick. Ipbrick.4cc. Disponível em: 26-06-2016. URL: http://www.ipbrick.com/products/ipbrick-4cc/.

[51] S.A. IPBrick. Ipbrick.vdi. Disponível em: 26-06-2016. URL: http://www.ipbrick.com/products/ipbrick-vdi/.

[52] IPBRICK S.A. *IPBRICK - 6.0 Version Installation Manual*. IPBRICK S.A., August 2014. Disponível em: 26-06-2016. URL: http://www.ipbrick.com/wp-content/uploads/2014/07/manual_inst_6.0_EN.pdf.

[53] Apple Support. Software update, upgrade–what's the difference?, march 2016. Disponível em: 26-06-2016. URL: https://support.apple.com/en-us/HT201564.

[54] Amazon Web Services. Amazon elastic compute cloud, 2016. Disponível em: 26-06-2016. URL: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-ug.pdf.

[55] Kuang Ning. Automate linux vm os updates using ospatching extension, October 2014. Disponível em: 26-06-2016. URL: https://azure.microsoft.com/en-us/blog/automate-linux-vm-os-updates-using-ospatching-extension/.

[56] Williamson Kevin. Role instance restarts due to os upgrades, September 2012. Disponível em: 26-06-2016. URL: https://blogs.msdn.microsoft.com/kwill/2012/09/19/role-instance-restarts-due-to-os-upgrades/.

[57] M G Pradeep. Windows azure iaas host os update demystified, november 2013. Disponível em: 26-06-2016. URL: https://blogs.msdn.microsoft.com/mast/2013/11/26/windows-azure-iaas-host-os-update-demystified/.

[58] Berjon (W3C) Robin, Doyle Navara (Microsoft) Travis, Leithead (Microsoft) Erika, O'Connor (Apple Inc.) Edward, e Pfeiffer Silvia. 1.9 a quick introduction to html, 2012. Disponível em: 26-06-2016. URL: https://www.w3.org/TR/2012/CR-html5-20121217/introduction.html#a-quick-introduction-to-html.

[59] Berners-Lee (MIT/W3C) T. e D. Connolly. Hypertext markup language - 2.0, 1995. Disponível em: 26-06-2016. URL: https://tools.ietf.org/pdf/rfc1866.pdf.

[60] Wium Lie Håkon. *Cascading Style Sheets*. Faculty of Mathematics and Natural Sciences - University of Oslo, 2000. Disponível em: 26-06-2016. URL: http://people.opera.com/howcome/2006/phd/.

[61] D. Cederholm e E. Marcotte. *Handcrafted CSS: More Bulletproof Web Design*. Voices That Matter. Pearson Education, 2010. Disponível em: 26-06-2016. URL: https://books.google.com/books?id=UgrUeIwsS60C.

[62] Lie (W3C) H., Bos (W3C) B., e Lilley (W3C) C. *The text/css Media Type*. The Internet Society, 1998. Disponível em: 26-06-2016. URL: https://tools.ietf.org/pdf/rfc2318.pdf.

[63] A. Meyer Eric. *Cascading Style Sheets: The Definitive Guide*. O'Reilly Media, 3rd edição, 2000. Disponível em: 26-06-2016. URL: https://tools.ietf.org/pdf/rfc2318.pdf.

[64] David Flanagan. *JavaScript: The Definitive Guide Activate Your Web Pages*. O'Reilly Media, Inc., 6th edição, 2011. Disponível em: 26-06-2016. URL: ftp://91.193.237.1/pub/docs/linux-support/programming/JavaScript/[O%60Reilly]%20-%20JavaScript.%20The%20Definitive%20Guide,%206th%20ed.%20-%20[Flanagan].pdf.

[65] Severance Charles. Javascript: Designing a language in 10 days. *Computer*, 45(2):7–8, 2012. Disponível em: 26-06-2016. URL: https://www.computer.org/csdl/mags/co/2012/02/mco2012020007.pdf, doi:http://doi.ieeecomputersociety.org/10.1109/MC.2012.57.

[66] A. Gutmans, S.S. Bakken, e D. Rethans. *PHP 5 Power Programming*. Bruce Perens' Open Source series. Prentice Hall PTR, 2005. Disponível em: 26-06-2016. URL: https://books.google.pt/books?id=oJ9QAAAAMAAJ.

[67] P. Litwin. Fundamentals of relational database design, 1995. Disponível em: 26-06-2016. URL: http://sbuweb.tcu.edu/bjones/20263/access/ac101_fundamentalsdb_design.pdf.

[68] The PostgreSQL Global Development Group. *PostgreSQL 9.5.2 Documentation*. The PostgreSQL Global Development Group, 2016. Disponível em: 26-06-2016. URL: https://www.postgresql.org/files/documentation/pdf/9.5/postgresql-9.5-US.pdf.

[69] Lorentz Diana e Beth Roeser Mary. *Oracle® Database SQL Language Reference*. Oracle, 11.2 edição, January 2016. Disponível em: 26-06-2016. URL: https://docs.oracle.com/cd/E11882_01/server.112/e41084.pdf.

[70] Ramakrishnan Raghu e Gehrke Johannes. *Database Management Systems*. McGrw-Hill Higher Education, 2nd edição, 1994. Disponível em: 26-06-2016. URL: http://202.74.245.22:8080/xmlui/bitstream/handle/123456789/608/databasemanagementsystems.pdf?sequence=1.

[71] Andrew Tridgell. *Efficient algorithms for sorting and synchronization*. Australian National University Canberra, 1999. Disponível em: 26-06-2016. URL: https://www.samba.org/~tridge/phd_thesis.pdf.