FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Infra-estrutura para a computação *multi-cluster* em ambiente *Grid*

Albano José Dias Serrano

Dissertação apresentada à Universidade do Porto
para a obtenção do grau de Mestre
(Área de especialização de Comunicações)

Orientador
Professor Doutor António Manuel da Silva Pina
Departamento de Informática da Universidade do Minho

Co-orientador
Mestre João Isidro Vila Verde
Faculdade de Engenharia da Universidade do Porto

Porto, Setembro de 2006

À memória do meu Pai

À Maria Antónia e ao Manuel

Resumo

As grades computacionais vieram desagregar utilizadores e aplicações de recursos físicos e domínios de administração. Até então, utilizadores estavam fortemente ligados aos *seus* sistemas de computação, enquanto as aplicações dependiam do domínio de administração que as geriam. As grades computacionais introduziram assim um nível de abstracção, separando utilizadores e aplicações das infra-estruturas computacionais e respectivas administrações. Estabeleceram uma nova organização, implementando plataformas de computação geograficamente distribuídas, aptas a servirem comunidades de utilizadores agrupados em Organizações Virtuais (VO), partilhando desta forma recursos. Contudo esta partilha não é simples. Uma das razões deve-se ao facto de frequentemente VOs requererem ambientes de execução diferenciados ou específicos.

Partindo de plataformas computacionais baseadas em *clusters* tipo *Beowulf*, consolidados, geograficamente distribuídos e integrados em diferentes domínios de administração, definiu-se como objectivo principal deste trabalho a constituição de uma plataforma trans-domínio que permitisse a execução de aplicações desenvolvidas em ambientes de programação por passagem de mensagens, perseverando os respectivos ambientes de execução originais, embora perspectivando uma utilização próxima da grade computacional.

Pela integração de várias tecnologias estabilizadas e desenvolvidas em código aberto foi possível implementar uma plataforma de computação alargada com uma arquitectura composta por três níveis: 1) rede virtual IP para interligação segura sobre redes públicas dos *clusters* participantes, fundada com tecnologia VPN com suporte TLS, projectada como uma rede em malha completa de túneis UDP associados em Domínios OpenVPN e garantindo total conectividade entre nós da plataforma, 2) sistema de ficheiros alargado suportado por extensões NFS que assegura a utilizadores e respectivas aplicações a partilha, entre todos os nós, de dados distribuídos pelo *multi-cluster*, 3) serviço de *clusters* virtuais que implementa, sobre a plataforma de computação heterogénea, ambientes de execução definidos dinamicamente pelo utilizador através de um portal associado ao serviço, cada um constituindo uma imagem de sistema uno com acesso interactivo através de sessão SSH iniciada pelo portal.

Os resultados obtidos permitem concluir que é viável a constituição de comunidades de utilizadores que partilham recursos de computação de múltiplos *clusters* distribuídos geograficamente, com o objectivo de rentabilizar a sua utilização, sem implicar alterações na arquitectura das suas plataformas locais de computação.

Futuramente, o serviço de *clusters* virtuais deve possuir mecanismos para descoberta e atribuição de recursos necessários à implementação de *clusters* definidos pelos utilizadores, assim como, é relevante o desenvolvimento de um *escalonador* que instancie, monitorize e termine *clusters* virtuais.

Abstract

Grid computing introduced an abstraction level that disaggregated users and applications from the computing physical resources and respective administration domains. Previously, users had to be strongly linked to their computing systems and applications depended on the administration domain for their management. Grid computing established new organizations that implement geographically distributed computation platforms capable of sharing resources between user communities associated in Virtual Organizations (VO). However, the fact that VOs frequently request differentiated or specific execution environments makes sharing hard.

With its foundation on consolidated, geographically distributed Beowulf clusters, with different administration domains, this work's purpose is the constitution of a trans-domain multi-cluster environment for the execution of application developed for message passing programming environments, aimed to maintain the original execution environments while, at the same time, endorsed the computing grid usage model.

Integration of several stable open source packages and technologies, lead to a computing platform architecture composed of three layers: 1) an IP virtual network for the secure integration of participant clusters over public networks, based on VPN technology with TLS support, designed as a full mesh network of UDP tunnels associated in OpenVPN domains and guaranteeing total connectivity between all the nodes of the platform; 2) a wide area file system supported by NFS extensions to allow users' data sharing between all nodes of the multi-cluster; 3) a virtual clusters (VC) service that implements a heterogeneous computing platform execution environment, dynamically defined by users through a portal associated to the service. Each VC constitutes a single system image with interactive and direct access via SSH session initiated through the portal.

The results of this work demonstrate the viability of constitution of multi-clusters with user communities sharing resources and maximizing utilization of multiple geographically dispersed clusters, without architectural changes in their local computing platforms.

In the future, we plan to implement a mechanism for the discovery and attribution of the necessary resources to create user defined clusters, as well as, to develop a scheduler that will support the activation, monitoring and termination of virtual cluster.

Prefácio

Perspectivei a pós-graduação em Redes e Serviços de Comunicação como uma oportunidade para validar e aprofundar conhecimentos adquiridos ao longo de uma carreira profissional nesta área, iniciada já nos *longínquos* finais dos anos 80. Mas também, como um investimento para encarar os desafios futuros que se colocam nesta disciplina tecnológica, cada vez mais presente na vida das pessoas e que está, e continuará, a revolucionar os nossos modos de vida.

A parte escolar da pós-graduação correspondeu às expectativas, apesar do esforço que representou para quem teve de dividir o tempo com a profissão, a família, as aulas, os trabalhos e os exames. Para a dissertação, devido a alterações na carreira profissional ocorridas entretanto, não se proporcionou a abordagem de temas que inicialmente tinha previsto.

Aceitei um tema proposto na área da computação distribuída com a convicção de que seria possível aplicar alguns conhecimentos de comunicações. Com o título "Infra-estrutura para a computação *multi-cluster* em ambiente Grid" a dissertação tinha como objectivo explorar a formação de uma plataforma computacional com base em *clusters Beowulf* que nos últimos anos se vulgarizaram, principalmente na comunidade académica. Com a conjunção destes recursos pretendia-se potenciar a sua rentabilidade mas também facultar um outro tipo de utilização, mais *distante* e cómoda, mais próxima da concepção de *Grid Computing*. Vários problemas se colocaram logo à partida e outros foram surgindo. No entanto, o objectivo foi atingido culminando na definição do conceito de *Cluster* Virtual e respectiva validação experimental.

Neste ambiente de computação as comunicações desempenham um papel fundamental, mas o trabalho revelou-se igualmente interessante pela interdisciplinaridade de conhecimentos a que obrigou, permitindo-me alargar horizontes, residindo aí a mais valia pessoal deste trabalho.

Agradeço aos meus orientadores: ao Prof. António Pina pela orientação aberta que conduziu proporcionando um diálogo permanente, pelo conhecimento e pelo incentivo que me transmitiu, pela sua disponibilidade; ao Mestre Isidro Vila Verde pela compreensão revelada e pela confiança que depositou em mim.

Agradeço muito à minha família, especialmente à Paula pelo apoio e encorajamento durante este percurso, pela sua tolerância e compreensão nos momentos mais críticos.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O tema deste trabalho relaciona-se directamente com a computação distribuída em ambientes de execução baseados em *clusters*. Neste capítulo, começa-se por fazer um enquadramento em relação a este tipo de recursos. De seguida é apresentada a motivação, identificando objectivos e problemas. Por último, e de forma resumida sugere-se a solução que permite alcançar os objectivos estabelecidos, contornando os problemas previstos.

1.1 Enquadramento

Este trabalho tem como ponto de partida plataformas de computação paralelas designadas por *clusters*. Importa assim, e desde já, caracterizar estas arquitecturas e registar a sua evolução até às grades computacionais, no sentido de se fornecer informação que ajude a compreender o trabalho e a sua motivação.

Clusters Beowulf

Foi ao longo dos anos 80 que a computação suportada em clusters adquiriu relevância, essencialmente como resultado da intensificação de três factores: a produção de processadores de alto desempenho, o desenvolvimento de redes de comunicação de baixa latência e a padronização de ferramentas para computação paralela e distribuída.

Por outro lado, a crescente capacidade de processamento exigida pelas aplicações e o elevado custo das arquitecturas paralelas dedicadas, tornando-as inacessíveis para uma grande maioria, motivou o desenvolvimento de plataformas computacionais baseadas em hardware comum, ou melhor, em componentes originalmente produzidos para outros fins, mas quando agrupados poderiam proporcionar o desempenho necessário para a execução de aplicações paralelas e distribuídas.

Em alternativa às arquitecturas específicas, como Cray e SGI, o grande desafio passava pela utilização de hardware de propósito geral combinado com software de código aberto de forma a construir-se soluções capazes de obter taxas de desempenho próximo das alcançadas pelos supercomputadores, mas como um custo bem mais baixo.

O projecto *Beowulf* [1] alcançou este desafio demonstrando a viabilidade de um *cluster*, *Beowulf*, como um computador paralelo construído com base em componentes físicos vulgares e que executa um sistema operativo e outro software igualmente vulgares. Um *cluster* é composto por nós – cada um contendo um ou mais processadores, memória que é partilhada por todos os processadores do mesmo nó, e dispositivos periféricos adicionais (como discos) – conectados por uma rede de comunicações que permite a troca de dados entre nós [2].

O aperfeiçoamento desta nova arquitectura paralela implicou investigação e desenvolvimento em áreas como comunicações rápidas, arquitecturas multi-processadores e em software eficiente e modular, capaz de extrair o máximo desempenho possível. Dado o grande número de aplicações já desenvolvidas foi ainda necessário adaptar protocolos e bibliotecas de programação existentes, tais como PVM e as implementações do padrão MPI.

A arquitectura de *cluster* procura integrar características presentes em três outras arquitecturas: redes de estações de trabalho (NOW), multi-processadores simétricos (SMP) e processadores maciçamente paralelos (MPP), visando a formação de uma plataforma de computação com funcionalidades essenciais para alto desempenho na execução de aplicações maioritariamente paralelas.

O desempenho desta arquitectura deve ser escalável, à medida que o número de nós de computação aumenta, sendo para isso fundamental uma utilização eficiente dos recursos. Assim, a existência de políticas eficientes de escalonamento e de mecanismos de balanceamento de carga são vitais para que um *cluster* com algumas dezenas ou centenas, ou mesmo milhares, de nós ofereça e sustente taxas elevadas de desempenho.

Independentemente da dimensão e do desempenho, as arquitecturas de cluster procuram implementar um sistema uno, um sistema de imagem única embora constituído por vários sistemas autónomos interligados. O conceito de *imagem única* dita que um sistema paralelo ou distribuído, independente de ser composto por vários processadores ou recursos geograficamente distribuídos, deve comportar-se com um sistema centralizado do ponto de vista do utilizador. Dessa forma, todos os aspectos relativos à distribuição de dados e de tarefas, comunicação e sincronização entre tarefas e a organização física do sistema devem ser abstraídos do utilizador, ou seja, devem ser transparentes para ele [3].

Neste sentido, ao longo destes anos desenvolveram-se vários ambientes de programação dedicados a diversos tipos de rede de comunicação (*Ethernet*, *Myrinet*, SCI), com diferentes funcionalidades e níveis de desempenho, muitos deles baseados nas bibliotecas PVM e MPI. Assim como, se desenvolveu um vasto conjunto de ferramentas de administração e escalonamento, tais como Scyld, Condor, Maui e PBS [4] com o objectivo de agregar todas as funcionalidades previstas para esse tipo de arquitectura.

A utilização de *clusters Beeowulf* como plataformas paralelas de alto desempenho multiplicaram-se ao longo dos anos 90 e teve o seu auge no final da década, com diversas instituições de ensino e comerciais adoptando este modelo de arquitectura para a investigação, desenvolvimento e execução de aplicações. A facilidade de construção, a eficiência na utilização dos recursos de hardware e o baixo custo, são alguns dos factores que contribuíram para o sucesso destas plataformas.

Em cerca de uma dezena de anos de investigação e desenvolvimento nesta área, diversas tecnologias foram propostas, diferentes ambientes de programação e protocolos de comunicação foram adaptados ou desenvolvidos e inúmeras aplicações foram testadas. Todo esse esforço resultou no estabelecimento crescente de *clusters* como plataformas alternativas para a execução de aplicações paralelas complexas. Como consequência, modelos de programação precisaram ser revistos, tecnologias e padrões consagrados na área de computação distribuída tiveram de ser substituídos por tecnologias mais apropriadas e um número cada vez maior de pessoas puderam ter

acesso à computação de alto desempenho [5].

Grades computacionais

De uma forma mais ou menos paralela ao estabelecimento de *clusters* como plataforma de alto desempenho, surgiu outra área de investigação e desenvolvimento denominada por *computação em grade* (*Grid Computing*) e que pode ser vista como a junção mais promissora entre a computação paralela e a computação distribuída.

Um sistema de Computação em Grade [6] pode ser definido de maneira bem abrangente como uma infra-estrutura de software capaz de interligar e gerir diversos recursos computacionais, possivelmente distribuídos por uma grande área geográfica, de maneira a oferecer ao utilizador acesso transparente a tais recursos, independente da localização dos mesmos. Na maioria dos casos, o principal recurso das grades é a capacidade de processamento, porém alguns sistemas incluem uma ampla gama de recursos como: dispositivos de armazenamento de grande capacidade, instrumentos científicos e até componentes de software, como bancos de dados. A origem do termo *Computação em Grade* deriva de uma analogia com a rede eléctrica (*Power Grid*), e reflecte o objectivo de tornar o uso de recursos computacionais distribuídos tão simples quanto ligar um aparelho na rede eléctrica.

O panorama actual apresenta inúmeros avanços na área da computação paralela, muitos deles impulsionados pelo advento dos *clusters* e tecnologias relacionadas a eles (protocolos de comunicação de alto desempenho, interfaces de rede programáveis, ambientes de gestão e monitorização de recursos, etc.). De um outro lado, pode-se destacar a crescente utilização da computação distribuída, principalmente a exploração da Internet e da Web – uma como infra-estrutura de comunicações e a outa como meio de navegação – para o desenvolvimento e execução de aplicações, bem como para a partilha e acesso a informação.

Um grade computacional pode ser vista então como uma tentativa de integração dessas duas áreas, em que a Internet e a Web podem ser usadas para permitir a partilha e acesso não só a informações geograficamente distribuídas, mas também a recursos computacionais e especializados, e ainda, uma maior interacção entre grupos de trabalho. Neste contexto, é cada vez maior o envolvimento de diferentes instituições para a definição de padrões, modelos de programação, arquitecturas básicas de software e políticas para que essa plataforma possa estabelecer-se como um alternativa atractiva para o desenvolvimento de novos tipos de aplicações, de maneira semelhante ao ocorrido com os *clusters*.

Um dos principais aspectos relativos à construção de plataformas baseadas em grades computacionais diz respeito à partilha de recursos em larga escala. Entretanto, outros factores devem ser considerados para a definição de um modelo estrutural para tal plataforma.

À semelhança das arquitecturas de *clusters* que integram características de outras arquitecturas com o objectivo de eficientemente proporcionarem altas taxas de desempenho na execução de aplicações paralelas complexas, também a arquitectura de grade computacional assimila aspectos de outras arquitecturas, podendo-se caracterizar estruturalmente uma grade como uma integração de recursos computacionais de alto desempenho (máquinas paralelas, *clusters* e instrumentos científicos) através de redes de comunicação de larga escala. Nesse contexto, uma estrutura de software para grade

deve fornecer funcionalidades que explorem eficientemente os recursos computacionais ao mesmo tempo que garantam o acesso confiável e seguro a esses recursos. Adicionalmente, a utilização de interfaces Web como principal forma de interacção com essas plataformas impõe algumas exigências e acrescenta alguma complexidade a essa estrutura de software, em favor da transparência ao utilizador final.

Uma arquitectura de grade computacional pode ser partilhada por diferentes Organizações Virtuais (VO), cada uma com características e necessidades específicas. É por isso errado pensar-se numa arquitectura padrão para grades computacionais, mas sim na definição de um conjunto básico de serviços que devem ser disponibilizados, possibilitando que cada VO implemente e acrescente essas funcionalidades da forma mais adequada [7].

Esse conjunto básico de serviços pode assim ser definido considerando que a concepção de grades computacionais, de uma forma geral, envolve o tratamento de algumas questões: 1) heterogeneidade de recursos geograficamente distribuídos, 2) escalabilidade de modo a suportarem um número elevado de recursos, 3) tolerância a falhas traduzida na capacidade de se adaptar aos recursos existentes de modo a manter desempenho e confiabilidade, 4) segurança de dados na medida que circulam em redes públicas e 5) mecanismos de gestão de aplicações capazes de atribuir, escalonar e contabilizar a utilização de recursos.

Apesar da diversidade e complexidade das grades computacionais é determinante para o seu desenvolvimento o estabelecimento de normas que especifiquem a arquitectura de implementação e a interligação de grades. Com este objectivo foi criado o *Global Grid Fórum* (GGF), um comité formado por representantes da investigação e indústria da computação em grade, no âmbito do qual foi definida a especificação: *Open Grid Services Architecture* (OGSA).

O OGSA [7] tem como objectivo definir conceptualmente os componentes de uma arquitectura de grade. Identifica os serviços essenciais para grades, definindo a sua finalidade e as suas inter-relações. Globus Tool Kit é uma implementação de grade computacional que inspirou a definição de OGSA e na qual se baseiam muitas grades de computacionais.

Deve-se ressaltar a importância do GGF [8] como fundamental para a ampla popularização e padronização das tecnologias de grade. Como se afirma em [6] no futuro, para que instituições façam parte de grades precisarão de implementar a arquitectura OGSA, tal como para fazer parte da Internet hoje é necessário utilizar o protocolo IP.

1.2 Motivação

É neste cenário, tendo em primeiro plano a arquitectura de *cluster* e em fundo a grade computacional, que se desenvolve este trabalho. Como se descreveu na secção Enquadramento, actualmente os *clusters* de alto desempenho, principalmente *Beowulf*, têm uma utilização muita alargada, quer na diversidade de aplicações quer no número de plataformas instaladas em diferentes instituições. São por isso recursos valiosos que justificam estratégias para maximizarem a sua rentabilidade, procurando ocupar todos os seus períodos ociosos.

É portanto natural que várias instituições, principalmente de investigação cientifica e de ensino, desenvolvam soluções que procurem tirar partido da junção das suas plataformas de computação baseadas em *clusters* de alto desempenho. Uma dessas soluções passa por implementar plataformas alargadas à semelhança das grades computacionais, como se abordou anteriormente. De facto é isso que acontece, havendo muitos exemplos de grades computacionais instaladas com base em implementações bem conhecidas: Globus [9], Condor, Legion, etc.

Este é um dos caminhos que um conjunto de instituições pode seguir para implementar uma infra-estrutura computacional comum, enriquecida pela heterogeneidade das diferentes plataformas individuais. Mas como se sabe as grades computacionais exigem dos utilizadores muitas vezes, outra forma de desenvolvimento de aplicações paralelas que contemple o novo ambiente de execução. Não se pode esquecer que o ambiente de execução de aplicações paralelas varia em diversos aspectos, dos quais se destaca conectividade, heterogeneidade, partilha de recursos, imagem do sistema e escalabilidade. Em que conectividade se refere aos canais de comunicação que interligam os processadores que compõem o ambiente de execução, heterogeneidade diz respeito às diferentes arquitecturas de processadores, partilha de recursos contempla a utilização destes entre várias aplicações, imagem do sistema corresponde a uma visão única do ambiente de execução e, por último, escalabilidade estabelece o número de processadores que integram o ambiente de execução.

Muitas aplicações paralelas já desenvolvidas contemplam ambientes de execução que não se adaptam a grades computacionais, sendo necessário proceder a alterações para que possam ser executadas numa plataforma do tipo grade, o que nem sempre é possível ou desejável.

É assim objectivo deste trabalho projectar uma solução que integre várias plataformas de *clusters Beowulf* geograficamente distribuídas e sob diferentes domínios de administração, que permita a execução de aplicações paralelas não-modificadas desenvolvidas em ambientes de programação PVM e MPI.

Deste enunciado se extrai que é importante disponibilizar às aplicações um ambiente de execução semelhante ao que é fornecido num ambiente de *cluster*. Isto por sua vez implica fundamentalmente com dois aspectos essenciais: 1) conectividade entre nós da *nova* plataforma de computação e 2) imagem de sistema uno, nomeadamente ao nível do sistema de ficheiros.

A conectividade envolverá questões relacionadas com a comunicação entre *clusters*, nomeadamente: 1) a interligação de redes privadas IP em que actualmente se baseiam este tipo de arquitecturas de computação, 2) a existência de sistemas de controlo de tráfego (*firewalls*) que limitam a comunicação trans-domínio e 3) a segurança de informação transmitida entre *clusters* sobre redes públicas.

Por sua vez a imagem de sistema uno abrangerá outro tipo de problemas, sabendo-se desde logo que nos *clusters Beowulf* se empregam sistemas de ficheiros baseados em NFS. Assim, surgem questões relacionadas com: 1) a partilha de dados e aplicações domínios de administração diferentes e 2) a gestão de acessos para uma comunidade alargada de utilizadores.

Tendo em conta que estão envolvidos *clusters* com domínios de administração diferentes estes aspectos assumem desde logo uma dimensão interessante ampliada por

um outro requisito externo: a nova plataforma deve perseverar a autonomia de cada *cluster* quanto à sua arquitectura e administração.

Adicionalmente ainda se considera importante que o acesso dos utilizadores aos recursos seja feito de uma forma cómoda, utilizando interfaces disponíveis em qualquer sistema operativo e a partir de qualquer localização.

1.3 Solução proposta

Com base na especificação e nos objectivos anteriormente definidos projecta-se uma solução descrita nos Capítulos 4 e 5, que responde aos problemas identificados integrando várias tecnologias apresentadas no Capítulo 3.

A proposta consiste na implementação de uma plataforma de computação alargada constituída como um somatório dos recursos constituintes dos *clusters* participantes na solução. Estabelecem-se três níveis:

- Rede virtual – rede de interligação garantindo a conectividade entre todos os nós independentemente do *cluster Beowulf* a que pertençam, implementada com base na tecnologia OpenVPN garante também a segurança entre *clusters*;
- Sistema de ficheiros alargado – implementa um sistema de ficheiros distribuído, utilizando procuradores NFS que compatibilizam os sistemas de ficheiros NFS dos *clusters*;
- Serviço de *cluster* virtual – permite a criação de *clusters* definidos pelo utilizador sobre a plataforma de computação alargada.

Com esta solução proposta obtém-se uma plataforma *multi-cluster* em que todos os *clusters* são ligados por rede de comunicações. Esta rede garante total conectividade TCP/IP entre todos os nós.

Dispondo desta rede de comunicações alargada a todos os nós, propõe-se a *uniformização* de todos os sistemas NFS dos *clusters* através da introdução de procuradores NFS que se encarregam de realizar o mapeamento dos atributos entre aqueles sistemas de ficheiros.

Com estes dois componentes, rede virtual e sistema de ficheiros alargados, obtém-se uma infra-estrutura computacional bastante abrangente, quer pela sua dimensão quer pela sua heterogeneidade, que fica disponível ao utilizador. Este terá então a oportunidade de, através de um portal Web, escolher entre os recursos disponíveis e de acordo com as suas necessidades, os recursos que integrarão o ambiente de execução das suas aplicações desenvolvidas em PVM ou MPI. O ambiente de execução assim obtido é considerado no âmbito deste trabalho um *Cluster* Virtual (CV) associado a um utilizador específico durante um determinado período de tempo. De retorno o Serviço de *Cluster* Virtual devolve ao utilizador via portal uma sessão SSH no *seu* CV.

Esta é a solução projectada tendo em conta os objectivos definidos e os problemas identificados. Para além da definição da solução o trabalho ainda consistiu na validação experimental de cada uma das suas componentes integradoras de várias tecnologias.

Nos capítulos seguintes é feita a explanação da solução. No capítulo 2 é feita a sua relação com outros trabalhos semelhantes. No capítulo 3 são apresentadas as principais tecnologias que integram a solução, enquanto nos capítulos 4 e 5 são especificadas as três componentes: rede virtual, sistema de ficheiros alargado e serviço de *cluster* virtual. Já no capítulo 6 são detalhados aspectos da implementação e, finalmente, no capítulo 7 são apresentadas as conclusões e perspectivado o trabalho futuro.

Capítulo 2

Trabalho relacionado

A apresentação da solução resultante deste trabalho, feita no capítulo anterior, dá a conhecer a constituição de uma plataforma de computação alargada, integrando *clusters Beowulf* de diferentes domínios de administração, vocacionada para executar aplicações paralelas não-modificadas em ambientes de execução configurados dinamicamente sobre a plataforma de computação, conforme especificações definidas pelo utilizador via portal Web. Os ambientes de execução assim obtidos tomam a designação de *Clusters* Virtuais, aos quais os respectivos utilizadores têm acesso directo e interactivo, como se de *clusters* reais se tratassem.

2.1 Execução de aplicações não-modificadas

Desta síntese destacam-se dois aspectos: execução de aplicações paralelas não-modificadas sobre uma plataforma de computação alargada.

A conciliação destes dois aspectos constitui o ponto de partida para este trabalho que encontra em *Running PVM Applications in the PUNCH Wide Area Network-Computing Environment* [12] aspectos comuns. De facto o trabalho referido procura estabelecer uma solução para executar aplicações desenvolvidas sob padrões de passagem de mensagens, como são o PVM e o MPI, na plataforma de computação PUNCH [13]. Tratando-se de uma grade computacional, PUNCH permite o acesso remoto a programas e recursos através de interface Web, não sendo assim viável o acesso directo às máquinas da plataforma.

Desta forma, PUNCH foi dotado de um mecanismo que suporta a execução de aplicações desenvolvidas em PVM ou MPI, consistindo no cumprimento de sete etapas. Na primeira o utilizador define via interface Web os requisitos necessários para a aplicação, especificando o número e o tipo de máquinas. A partir desse momento o PUNCH desencadeia as restantes acções. Primeiro, procedendo à selecção e à reserva de recursos, que são feitas por um gestor de recursos. Os recursos disponíveis estão organizados em conjuntos por domínio, sendo sempre atribuídos recursos de um só domínio. As etapas seguintes continuam a ser realizadas automaticamente, passando pela criação do ambiente de execução PVM e quando este está pronto a aplicação é iniciada em modo de lote ou em modo interactivo pelo utilizador. Finalmente os resultados são disponibilizados via portal do PUNCH.

A componente Serviço de *Cluster* Virtual (SCV) que integra este trabalho de dissertação difere pela positiva em relação ao PUNCH em três aspectos: 1) a selecção dos recursos é feita directamente pelo utilizador com base nos recursos disponibilizados pelo SCV no portal, 2) o ambiente de execução pode ser constituído por nós de

computação de diferentes domínios tirando partido da heterogeneidade de recursos, tal é possível devido à conectividade total entre todos nós implementada pela rede virtual e ao sistema de ficheiros alargado que permite a formação de um sistema de ficheiros comum aos nós do ambiente de execução, 3) após constituído dinamicamente o ambiente de execução, o utilizador tem acesso directo e em modo interactivo a todos os nós que constituem o ambiente de execução, designado por *Cluster* Virtual no âmbito deste trabalho.

2.2 *Clusters* a pedido

O desenvolvimento do trabalho levou à definição do conceito de *Cluster* Virtual como uma abordagem intrínseca ao próprio trabalho. Mais tarde constatou-se na bibliografia a existência desta designação e com um significado muito aproximado.

No âmbito do presente trabalho, *Cluster* Virtual significa um ambiente de execução definido pelo utilizador com base nos recursos disponíveis na plataforma *multi-cluster*, formando dinamicamente um ambiente semelhante ao de um *cluster*: um nó principal e vários nós de computação partilhando entre si um sistema de ficheiros uno.

Na bibliografia, entretanto consultada, verificou-se a existência de projectos que empregam também a designação de *Cluster* Virtual (*Virtual Cluster*) ou *Cluster* a Pedido (*Cluster on Demand*). Nestes casos referem-se a *clusters* igualmente especificados pelos utilizadores e também construídos dinamicamente, mas com base na tecnologia de máquinas virtuais.

As grades computacionais vieram desagregar utilizadores e aplicações de recursos físicos e domínios de administração. Até então, utilizadores estavam fortemente ligados aos *seus* sistemas de computação, enquanto as aplicações dependiam do domínio de administração que as geriam. As grades computacionais introduziram assim um nível de abstracção, separando utilizadores e aplicações das infra-estruturas computacionais e respectivas administrações. Estabeleceram uma nova organização, implementando plataformas de computação geograficamente distribuídas aptas a servirem comunidades de utilizadores agrupados em Organizações Virtuais (VO), partilhando desta forma recursos. Contudo esta partilha não é fácil. Uma das razões deve-se ao facto de frequentemente VOs requererem ambientes de execução diferenciados ou específicos.

Actualmente, as grades computacionais evoluem, entre outros, no sentido da introdução de outros níveis de abstracção, em grande parte motivadas pelo desenvolvimento das tecnologias de virtualização de hardware, lideradas por soluções como VMware e Xen.

De facto a combinação de máquinas virtuais com a tecnologia de *Grid* Computacional potencia soluções para grande parte dos problemas que afectam a computação distribuída, nomeadamente: vulnerabilidades de segurança, sobrecarga administrativa, compatibilidade entre aplicações e recursos.

A utilização de máquinas virtuais proporciona a criação de *clusters virtuais* [14], sendo *cluster* virtual um conjunto de máquinas virtuais configurado para funcionar como *cluster* e posteriormente integrado numa grade. Este *novo* recurso tem a vantagem de poder ser configurado de acordo com os requisitos de uma determinada comunidade

de utilizadores, por exemplo uma VO. Esta vantagem toma ainda uma dimensão maior quando a especificação e instalação são feitas de forma dinâmica, quando utilizadores através de interfaces Web podem definir *a pedido* a configuração de *clusters* virtuais e de seguida desencadear os processos de instalação e activação via uma camada de software desenvolvida para esse efeito.

Os *clusters* virtuais correspondem por agora a uma abstracção do nível físico das grades computacionais com o objectivo de resolver alguns dos problemas acima referidos, libertando escalonadores trans-domínio de tarefas relacionadas com: o processo de atribuição de recursos a aplicações, os recursos físicos e as configurações locais de software. O nível de abstracção introduzido por *clusters* virtuais permite mapear aplicações ou trabalhos para recursos virtuais, os quais, por sua vez, são geridos por domínios e ambientes físicos.

É neste contexto que se desenrolam alguns projectos, dos quais se destacam In-VIGO e COD. A plataforma In-VIGO [15] [16] fornece um ambiente distribuído onde múltiplas instâncias de aplicações podem coexistir em recursos virtuais ou físicos, de modo transparente para os utilizadores. Neste projecto é utilizada a tecnologia de virtualização para criar conjuntos de recursos virtuais que podem ser configurados e agregados a pedido dos utilizadores, o que implica uma gestão dinâmica destes recursos.

COD (*Cluster-on-Demand*) [17] identifica os problemas associados à partilha de *clusters Beowulf* consolidados em plataformas de *Grid*. Neste sentido, COD propõe-se gerir um *cluster* como um recurso modular multi-propósito que suporta diferentes grupos de utilizadores e diferentes ambientes de software, isolados em partições designadas *clusters virtuais*. COD apresenta-se como um gestor dinâmico de recursos de *cluster*, atribuindo servidores físicos disponíveis num lote a diversos *clusters* virtuais configurados de forma independente com software, espaço de nomes, contas de acesso e volumes de armazenamento em rede. Este gestor implementa um conjunto de funcionalidades que permite a formação de *clusters* virtuais dinamicamente.

Numa configuração em grade, COD funciona como um gestor local que controla os recursos locais e exporta uma interface de negociação de recursos para um serviço intermédio da grade, o qual faz a ligação à plataforma da grade. Neste contexto, *clusters* virtuais são a base para uma política dinâmica de atribuição de recursos para diferentes ambientes de software e comunidades de utilizadores. A implementação de *clusters* virtuais dinâmicos corresponde a uma abstracção para gestão avançada de recursos em ambientes de grade computacional.

Estas abordagens e o SCV proposto neste trabalho partilham o objectivo de dotarem plataformas de computação de mecanismos que permitam corresponder a utilizadores ou aplicações específicos, facilitando a definição dinâmica de ambientes de execução. No SCV esta particularidade consegue-se tirando partido da heterogeneidade da plataforma *multi-cluster*. Quanto mais esta for diversificada, maior será a capacidade de gerar ambientes de execução diferenciados. No caso dos projectos apresentados os ambientes de execução diferenciados obtém-se pela utilização de máquinas virtuais dinamicamente definidas e instaladas sobre um conjunto de recursos físicos reservados para esse fim. Este processo torna-se mais complexo e implica a construção de plataformas já com este propósito. A vantagem do SCV é que não introduz alterações às arquitecturas dos *clusters* participantes no serviço.

Capítulo 3

Tecnologias

Neste capítulo é feita uma descrição das principais tecnologias que integram a solução projectada no âmbito deste trabalho. Não se pretende uma abordagem exaustiva sobre cada uma das tecnologias, mas apenas com detalhe suficiente para que se disponha de informação necessária à compreensão da sua utilização no contexto da solução onde são utilizadas. Começa-se pela tecnologia de "máquinas virtuais" que permite simular a componente física para suportar o ambiente *multi-cluster*, enquanto na secção "plataforma de *cluster*" se aborda a componente lógica para o mesmo ambiente de desenvolvimento. Nas duas secções seguintes são apresentadas as duas tecnologias que integram as soluções "plataforma *multi-cluster*" e "serviço de *cluster* virtual", respectivamente, "rede privada virtual" e "procurador NFS".

3.1 Máquinas Virtuais

Aborda-se nesta primeira secção a tecnologia "máquinas virtuais" que permite com reduzidos recursos construir um cenário suficientemente *real* para a realização deste trabalho.

3.1.1 Introdução

Há muito que é familiar o conceito de máquina virtual, como sendo software que cria um novo ambiente entre a plataforma física e o utilizador final, onde, por sua vez, se pode executar aplicações.

Enquanto uma máquina real é um computador real, isto é, feito de componentes físicos e executando um sistema operativo específico, uma máquina virtual é um computador fictício criado por um programa de emulação. Os seus componentes, memória, processador e outros recursos, são virtuais.

Uma categoria de máquinas virtuais consiste em computadores fictícios desenvolvidos com uma finalidade muito específica, como é o caso da JVM (*Java Virtual Machine*), para a qual existem simuladores nas mais diversas plataformas, desde computadores de grande porte até telemóveis, tornando as aplicações Java extremamente portáveis.

Outra categoria de máquinas virtuais é constituída pelos emuladores que simulam computadores reais. Na actualidade destacam-se os emuladores de PCs como o VMware, o Virtual PC da Microsoft ou o Xen.

Neste trabalho usaram-se os dois tipos de máquinas. O JVM como parte integrante da solução e o VMware com ferramenta de apoio ao desenvolvimento do projecto. De imediato aborda-se o VMware quanto ao seu funcionamento e contexto de utilização.

3.1.2 VMware

As tecnologias de máquinas virtuais tornaram-se numa forma bem aceite de virtualização. Essencialmente, estas tecnologias permitem hospedar numa única máquina física múltiplas instâncias virtuais de sistemas operativos, permitindo desta forma que sistemas operativos diferentes, tais como Solaris, Linux e Windows coexistam, sem interferência, no mesmo hardware.

A virtualização de infra-estruturas fornece abstracção de processamento, armazenamento de dados, rede e software eliminando desta forma as restrições físicas, uma vez que todos esses meios ficam acessíveis como agrupamentos lógicos, possíveis de alocar quando e onde necessário.

Nos últimos anos, a tecnologia VMware evidenciou-se como um destacado fornecedor de soluções de máquinas virtuais. A tecnologia de virtualização VMware [18] [19] baseia-se num supervisor que funciona como um mini-sistema operativo que é executado em primeiro lugar e, essencialmente, virtualiza o hardware subjacente em múltiplas plataformas para que os diferentes sistemas operativos possam correr no mesmo servidor físico. Esta abordagem provou ser particularmente atractiva para as organizações com aplicações que exigem um ambiente de sistemas operativos misto.

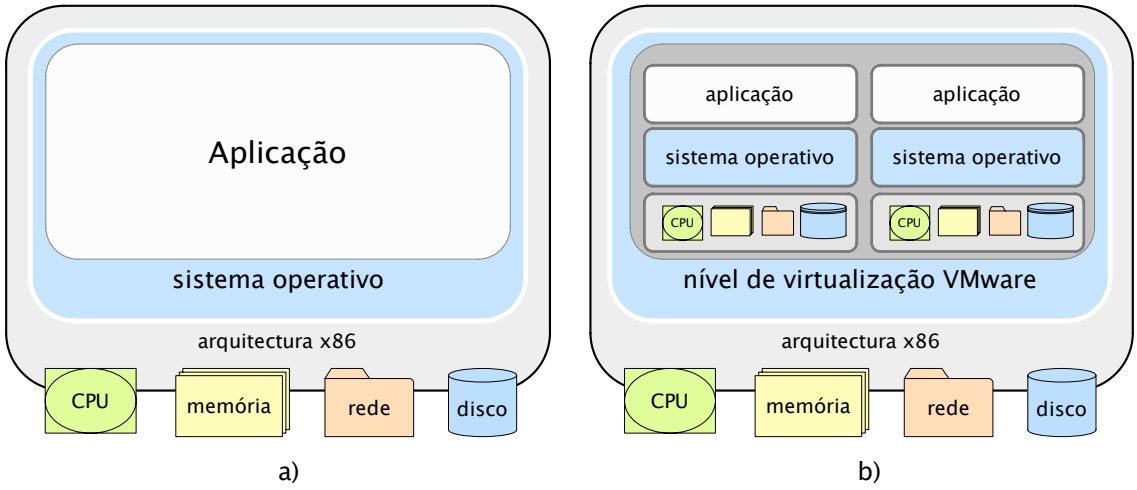

Figura 3.1 – Virtualização de plataforma pelo VMware: a) antes b) depois [18].

Na Figura 3.1 representa-se uma plataforma baseada na arquitectura x86, antes e depois da virtualização com tecnologia VMware. Antes da virtualização dispunha-se de: 1) uma única imagem de sistema operativo por máquina – software e hardware estavam intrinsecamente ligados, 2) a execução de aplicações múltiplas geravam frequentes conflitos, 3) era impossível reutilizar recursos disponíveis numa infra-estrutura rígida que, em consequência de todos estes factores, se torna cara.

Com o nível de virtualizição introduzido pelo software VMware obtém-se: 1) independência entre o hardware e o sistema operativo e as aplicações, 2) é possível ter várias máquinas virtuais suportando diferentes sistemas operativos, 3) cada sistema operativo e cada aplicação são geridos como uma única unidade por encapsulamento de ambos numa máquina virtual. Este modelo de virtualização é conhecido como fraccionamento.

Para sistemas Unix/RISC e sistemas x86, as duas abordagens típicas usadas para o fraccionamento baseado em software são as arquitecturas de hospedagem e de supervisor, conforme se ilustra na Figura 3.2.

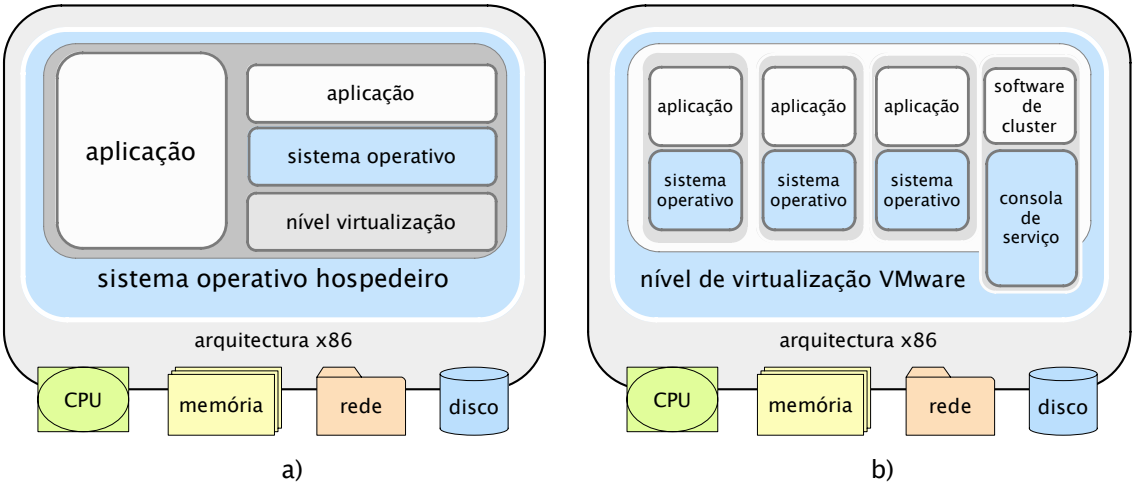

Figura 3.2 – Arquitecturas de fraccionamento: a) hospedagem e b) supervisor.

Na arquitectura de hospedagem o fraccionamento é feito sobre o sistema operativo normal, suportando uma variada gama de hardware. Na arquitectura de supervisor, este constitui o primeiro nível de software instalado sobre o hardware x86, e uma vez que tem acesso directo aos recursos físicos, acaba por ser mais eficiente que arquitectura de hospedagem, permitindo escalabilidade, robustez e desempenho. O supervisor funciona como um mini-sistema operativo que é executado em primeiro lugar e, essencialmente, virtualiza o hardware subjacente em pequenas partes, para que os diferentes sistemas operativos possam correr no mesmo servidor físico. Os supervisores podem ser desenhados para serem fortemente ligados a sistemas operativos ou, pelo contrário, para serem independentes.

O VMware implementa as duas arquitecturas nas suas soluções, das quais se destacam: a solução GSX que funciona sobre sistemas operativos Linux e Windows XP – virtualização por hospedagem – e a solução ESX que trabalha directamente sobre plataformas de hardware certificadas pela própria VMware – virtualização por supervisor. A primeira tem a vantagem de funcionar sobre qualquer computador vulgar com os requisitos mínimos para suportar os sistemas operativos hospedeiros.

3.1.3 Contexto de utilização

Para a realização deste projecto era necessário dispor de um conjunto de *clusters* que permitissem a realização faseada de ensaios, destinados a comprovar as soluções progressivas e parciais, essenciais para definir uma implementação da solução global.

Obviamente que dada a natureza dos ensaios não seria viável utilizar *clusters* em produção, haveria certamente interferência no regular funcionamento, prejudicando dessa forma trabalhos de utilizadores em execução. Por outro lado, a constituição de uma plataforma com os requisitos mínimos para a realização do trabalho corresponderia sempre a esforço considerável, principalmente logístico. Estes argumentos justificaram assim a utilização de tecnologia de máquinas virtuais para a construção de um *multi-cluster* de desenvolvimento.

Desta forma definiu-se a utilização do software VMware GSX como plataforma de virtualização do ambiente *multi-cluster*. Outras soluções de virtualização, como o XEN [20], foram consideradas mas a opção pelo VMware justificou-se essencialmente pela experiência de utilização já existente.

Concretamente, este *laboratório virtual* assentou num sistema constituído por um processador Intel Pentium 4 a 3.0 GHz, 2 GB de memória principal, disco rígido com 80 GB, unidades de CD-ROM e disquete 1,44 MB, tendo como sistema operativo o Linux Red Hat 9.0.

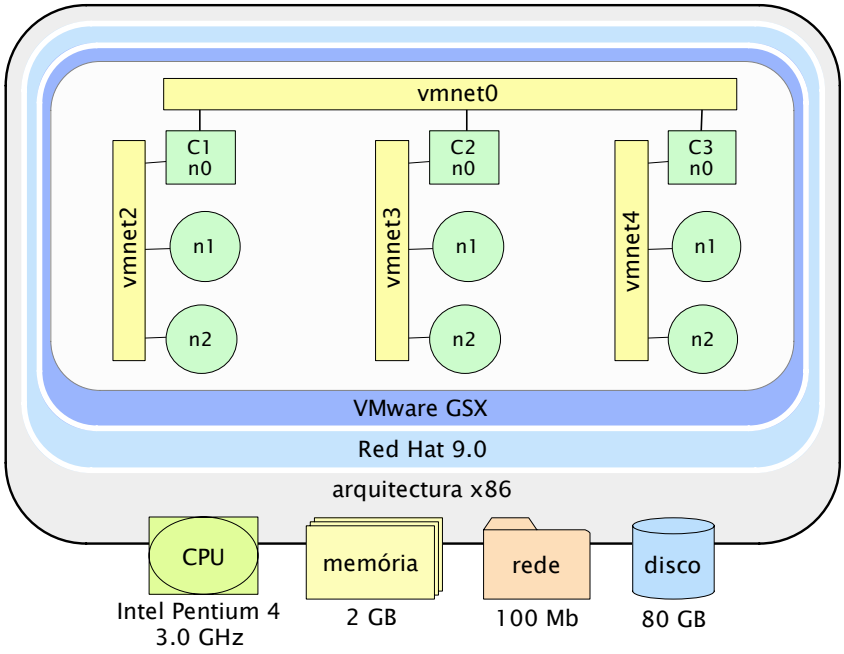

Figura 3.3 – Plataforma de desenvolvimento virtual.

Instalado o software VMware GSX 3.30 procedeu-se à instalação e configuração de três *clusters*, cada um correspondendo a uma arquitectura *Beowulf* típica com três máquinas virtuais: nó frontal e nós de computação, os quais se interligam através de uma rede IP privada, tendo como *gateway* para o exterior o nó frontal que se encontra

ligado a uma outra rede para simular uma rede pública. Na Figura 3.3 apresenta-se a plataforma virtual criada para o desenvolvimento do projecto constituída por:

- *Cluster* C1 – nós: C1-n0, C1-n1, C1-n2; rede privada: vmnet2;
- *Cluster* C2 – nós: C2-n0, C2-n1, C2-n2; rede privada: vmnet3;
- *Cluster* C3 – nós: C3-n0, C3-n1, C3-n2; rede privada: vmnet4;
- Rede de interligação, equivalente à rede pública: vmnet0.

A instalação e a configuração de cada *cluster* consistiram, resumidamente, nos seguintes procedimentos:

- Criação das máquinas virtuais especificando o respectivo hardware virtual:
- Nó frontal: 256MB de memória, 8 GB, de disco, atribuídos dinamicamente, em blocos de 2GB, duas placas Ethernet, unidade de CR-ROM e sistema operativo Linux Red Hat;
- Nós de computação: 128MB de memória, 4 GB de disco, atribuídos dinamicamente, em blocos de 2GB, uma placa de rede e sistema operativo Linux Red Hat;
- Instalação do sistema de *cluster* NPACI Rocks no nó frontal, de acordo a estratégia definida mais à frente na secção "Plataforma de *Cluster*", seguindo os procedimentos de instalação e configuração sugeridos por este sistema;
- Instalação dos nós de computação a partir do nó frontal através de rede.

Concluídos estes procedimentos para cada *cluster*, obteve-se um *multi-cluster* constituído por três *clusters*, neste caso homogéneos, funcionando sobre uma infra-estrutura virtualizada pelo VMware GSX. A partir daqui todo o desenvolvimento do projecto evolui como se tratasse de uma plataforma física. Apenas por limitação de recursos, processador e memória, não se utilizaram todos os nós em simultâneo.

Em conclusão, o recurso à tecnologia de máquinas virtuais foi determinante para a realização deste projecto, pela reduzida afectação de recursos, e consequente redução de custos, e ainda pela facilidade de comodidade de utilização e experimentação.

3.2 Plataforma de *cluster*

Apresenta-se aqui uma componente utilizada no trabalho desenvolvido que não sendo essencial para a solução projectada, constitui o ponto de partida. De facto pretende-se implementar uma infra-estrutura alargada de computação alicerçada numa plataforma de *cluster* consolidada, robusta, flexível, fácil de gerir.

3.2.1 Introdução

Duas vezes por ano, um grupo de fabricantes, cientistas de computação, peritos em computação de alto desempenho (HPC) e elementos da comunidade Internet compilam e difundem uma lista dos 500 sistemas de computação mais poderosos no mundo. Nos últimos cinco anos, a percentagem de *clusters* no "TOP500 Supercomputer Sites" [21]

cresceu de 2.2 por cento para 60.8 por cento. Durante o mesmo período de tempo, a dimensão típica destes clusters cresceu de alguns centenas de processadores para milhares de processadores. Actualmente, grandes *clusters* baseados em processadores Intel ou ADM incluem mais de 10,000 processadores [22].

O desenvolvimento, manutenção, monitorização, e administração de *clusters* podem tornar-se processos complexos e demorados. Antes de uma aplicação utilizar o elevado poder de computação disponível num *cluster* de alto desempenho, os administradores têm de o instalar, de o configurar e posteriormente proceder à sua monitorização e administração. Manualmente, a implementação mesmo de um pequeno *cluster* de alto desempenho não é uma tarefa fácil. O trabalho necessário para colocar um nó de *cluster* funcional, com o seu sistema operativo devidamente instalado e configurado, implica instalar múltiplos componentes, redes, bibliotecas paralelas e software de administração.

Uma solução de *cluster* desejável consiste numa ferramenta semi-automática que apoie os administradores a realizar as tarefas de desenvolvimento, manutenção e administração de *clusters*, principalmente, de alto desempenho. A complexidade de administração de um *cluster*, por exemplo, garantir que todos os nós têm um conjunto consistente de software, leva com frequência os administradores, muitas vezes investigadores sem conhecimentos na área, também utilizadores do *cluster*, a um de dois extremos: o *cluster* não é estável devido a problemas de configuração, ou o software fica desactualizado com falhas de segurança e com erros de software conhecidos mas permanecendo muito tempo sem serem corrigidos.

É neste contexto que é relevante a utilização de uma plataforma de *cluster*, como sendo uma ferramenta essencial para oferecer aos utilizadores *clusters* fiáveis, com alta disponibilidade e desempenho. De entre algumas soluções, opta-se por usar neste trabalho, como base da solução global a plataforma de gestão de *clusters* Rocks.

3.2.2 Rocks

A plataforma Rocks foi desenvolvida em 2000 pela Sociedade Nacional para Infra-estrutura de Computação Avançada dos Estados Unidos (NPACI - *National Partnership for Advanced Computational Infrastructure*) como uma ferramenta para simplificar a administração de *clusters*. Desde então, tem-se tornado uma solução *de facto* para a gestão destes sistemas. Em Outubro de 2005 os utilizadores registados do NPACI Rocks representavam um poder de cálculo avaliado em 149 triliões de operações em virgula-flutuante por segundo (TFLOPS) [22].

O Rocks baseia-se na distribuição de Linux Red Hat e em vários pacotes de software de construção de *cluster*, actualmente todos fornecidos no formato *Rolls*. Exemplos de *Rolls* que integram o Rocks são: base, Kernel, HPC, OS, entre outros.

Considera-se relevante descrever de seguida, de forma sucinta, a arquitectura e o funcionamento desta plataforma de gestão, bem como de alguns serviços importantes que a integram.

Arquitectura

A Figura 3.4 apresenta a arquitectura tradicionalmente usada em *clusters* de alto desempenho, popularizada pelo projecto *Beowulf* e que é adoptada pelo Rocks. Trata-se de uma arquitectura que favorece um elevado número de componentes de baixo custo, que sendo fáceis de substituir garantem uma maior disponibilidade.

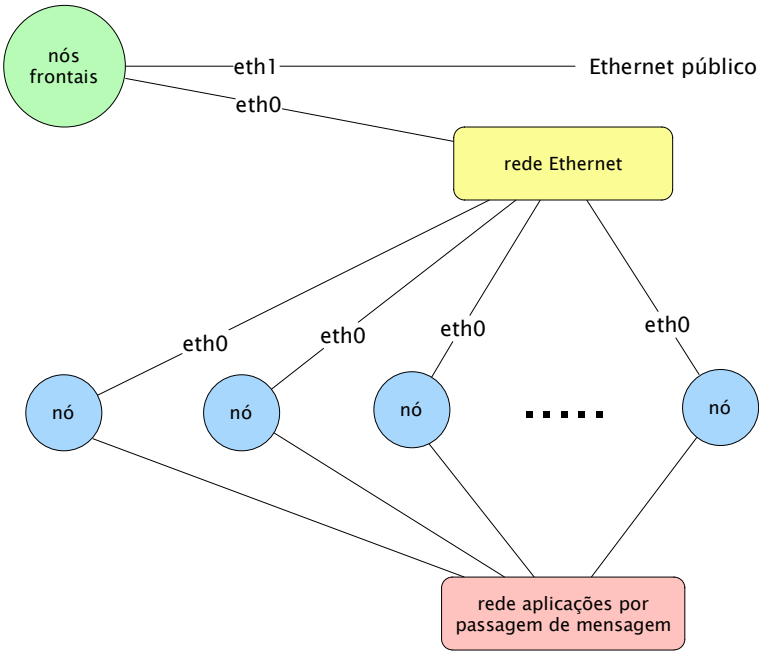

Figura 3.4 – Arquitectura típica de um *cluster* Rocks [23].

Nesta arquitectura identificam-se dois tipos de nós: frontais e de computação. Os nós frontais são instalados com software amplamente utilizado no desenvolvimento de aplicações de *cluster* e na execução de aplicações paralelas. Nomeadamente, compiladores Fortran e C/C++ da Intel, bibliotecas *Math Kernel* também da Intel, assim como várias bibliotecas para ambientes de programação paralela, como MPICH e PVM, ambos com suporte para sistemas de comunicação *Myrinet* e *Ethernet*. Para suportar a execução de trabalhos em ambientes de produção, são utilizados os escalonadores PBS (*Portable Batch System*) e Maui. Em termos de serviços, os nós frontais fornecem, entre outros, DNS, DHCP, NAT e serviço de *firewall* implementado com o software IPtables. Os utilizadores acedem ao *cluster* através dos nós frontais usando clientes SSH.

Os nós de computação são sistemas dedicados à execução de trabalhos submetidos pelos utilizadores. O máximo desempenho é ditado pelo número e tipo de nós de computação, os quais são em regra em maior número que os nós frontais. Assim sendo, o Rocks prevê a existência de vários tipos de nós de computação. Estes nós são instalados automaticamente a partir dos nós frontais, através de um processo bastante transparente para o administrador. Aliás, a instalação é um ponto forte do Rocks. Um único nó frontal pode instalar simultaneamente mais de uma centena de nós de computação em poucos minutos.

Um dos principais objectivos da administração de *clusters* é garantir a consistência do software de cada nó. Enquanto ferramentas mais antigas consomem muitos recursos

a comparar e actualizar versões de software, a estratégia do Rocks passa essencialmente por proceder à reinstalação de cada nó de computação sempre que é detectada alguma inconsistência. Como o processo de instalação é simples e rápido este método torna-se bastante eficiente.

Na verdade, a estratégia de administração do Rocks assenta no seguinte principio: deve ser fácil instalar qualquer versão de software em qualquer nó do *cluster*, independentemente da dimensão deste. Para implementar esta visão, define-se as seguintes regras: 1) todo o software integrado em *clusters* Rocks está em RPMs (*Red Hat Package Manager)*, 2) a configuração dos nós de computação é totalmente automática, 3) é essencial usar serviços escaláveis (HTTP, NIS, etc.) [24].

A implementação destas regras é suportada por duas tecnologias desenvolvidas pelo Red Hat e que são fundamentais no Rocks: RPM e *Kickstart*. Enquanto um pacote RPM contém todos os ficheiros, desde os binários até aos manuais, necessários para a instalação de um determinado módulo de software, *Kickstart* automatiza a instalação de pacotes de software nos nós de acordo com um ficheiro de configuração que contém o nome dos pacotes a instalar, os comandos a executar e as respostas a todas as perguntas efectuadas numa normal instalação interactiva de um nó.

O Rocks utiliza as potencialidades do *Kickstart* para fazer a configuração totalmente automática dos nós de computação. Caracterizando-se estes nós por utilizar um sistema operativo sem estado, e requerendo a configuração totalmente automática, torna-se desta forma sustentável a expansão de um *cluster*, ao ponto de, por cada nó de computação que lhe é acrescentado, corresponder apenas um incremento muito reduzido na administração global do sistema.

O requisito da escalabilidade passa pela utilização de serviços escaláveis e serviços dinâmicos que comuniquem com frequência aos nós de computação a mudança de estado, nomeadamente da informação de utilizadores, configuração de rede, etc. Na instalação dos nós de computação usa-se PXE (*Pre-execution Environment*) para obter, a partir do nó frontal, um sistema operativo mínimo para o arranque dos nós, depois do qual se utiliza o método HTTP do *Kickstart* para descarregar RPMs através da rede. Estes são depois instalados automaticamente. Na configuração das interfaces Ethernet dos nós de computação é utilizado o DHCP (*Dynamic Host Configuration Protocol*).

Embora não sendo um serviço escalável, é ainda usado NFS (*Network File System*), para partilhar entre os nós de computação as directorias de trabalhos dos utilizadores definidas no nó frontal.

Finalmente, o software de máquinas em produção deve ser sistemática e continuamente actualizado. Para isso o Rocks disponibiliza a ferramenta Rocks-dist que facilmente actualiza uma distribuição Rocks. Esta ferramenta pode ser utilizada para se proceder a actualizações de segurança e corrigir erros de software.

Deve-se ainda referir outros componentes importantes do Rocks: serviço 411, base de dados *Cluster* e o sistema de monitorização Ganglia [25].

Serviço de informação

O serviço 411 é semelhante ao NIS (*Network Information System*), tendo substituído este a partir da versão 3.1 do Rocks. É usado para distribuir de forma segura

ficheiros de senhas, ficheiros de configuração de utilizadores, grupos e outros ficheiros de configuração. Utiliza criptografia de chave pública para proteger o conteúdo dos ficheiros. Opera ao nível do ficheiro, não usando RPCs como o NIS, e distribui os ficheiros via HTTP.

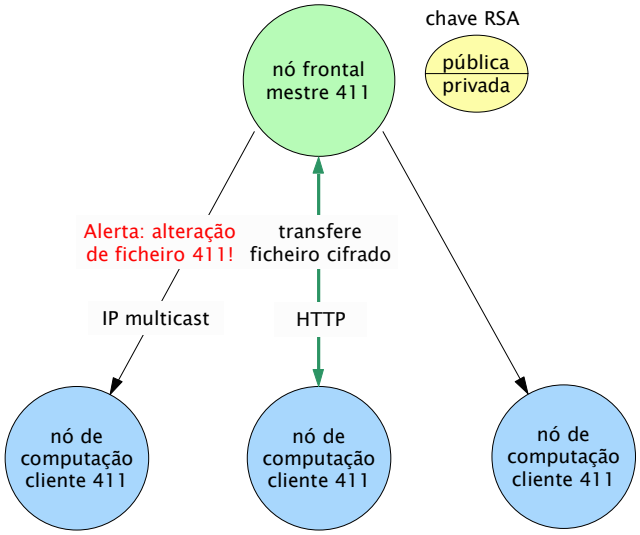

Figura 3.5 – Arquitectura do serviço 411 [23].

O serviço 411 define um nó mestre e nós clientes. Os nós mestres encriptam e disponibilizam ficheiros 411 usando os seus servidores *Apache*. Os nós clientes, normalmente os nós de computação, obtêm os ficheiros 411 usando HTTP, decifrando-os e guardando-os depois nos sistemas de ficheiros locais. Na Figura 3.5 ilustra-se a situação em que o ficheiro com as contas de utilizadores foi alterado no nó frontal. O 411 envia um alerta em *multicast* UDP para os nós de computação, notificando a alteração. Por sua vez, os nós de computação que receberam a mensagem transferem o ficheiro alterado. Um alerta é reenviado várias vezes para que se garanta que todos os nós tiveram conhecimento da alteração.

O Rocks constrói automaticamente uma base de dados suportada em MySQL que contém informação sobre a componente física e sobre as aplicações instaladas no *cluster*. A base de dados designa-se por "*Cluster*" e contém as seguintes tabelas: aliases, app_globals, appliances, distributions, memberships, networks, nodes e versions. Esta base de dados permite ter toda a informação sobre a configuração do *cluster* num formato uniformizado, a partir da qual se geram dinamicamente ficheiros de configuração necessários ao sistema operativo e aplicações. A base de dados é assim uma componente crucial do sistema Rocks.

Serviço de monitorização

Ganglia é o sistema de monitorização adoptado pelo Rocks. É um sistema distribuído, baseado em comunicação *multicast*, tendo cada nó um agente (designado por *dendrite*). Cada um destes agentes colecta configurações de hardware e software, alterações de estado, e informação sobre o estado dos nós. Todas estas informações são

enviadas para outros agentes (designados por *axons*) que interrogados através de comandos do Ganglia dão conta do estado do *cluster*.

Periodicamente, uma vez por hora, todos os agentes *dendrites* comunicam informação sobre configurações estáticas, como: número de CPUs e respectivas velocidades, versão do *Kernel* do SO e a quantidade de memória RAM instalada. De 5 em 5 segundos, os mesmos agentes informam sobre alterações significativas em termos de tempo de CPU, médias de carga, memória em uso e número de processos em execução.

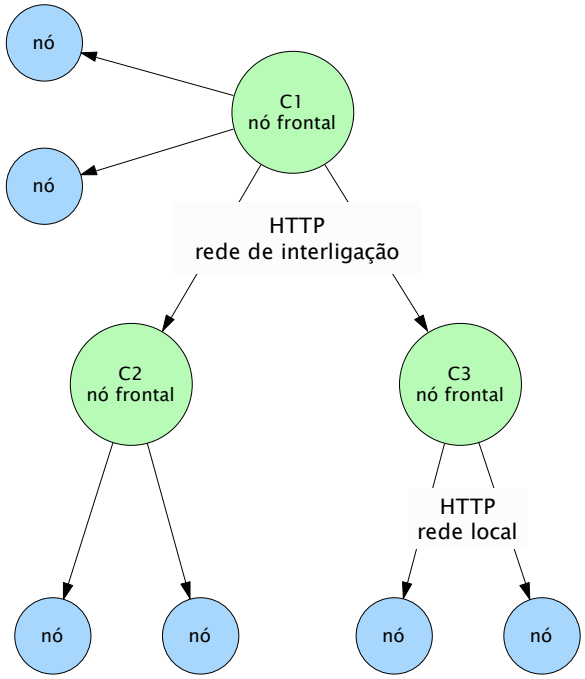

Figura 3.6 – Instalação do Rocks na plataforma de desenvolvimento.

3.2.3 Contexto de utilização

Como já referiu em 3.1.3 o *multi-cluster* de desenvolvimento é instalado sobre máquinas virtuais, sendo composto por três *clusters* Rocks. Cada um composto por um nó frontal e dois nós de computação.

A versão do Rocks é a 3.30 e a instalação necessária para o trabalho consiste somente nos CDs "Rocks-Disk1" e "Roll-HPC+Kernel" [26]. Como se referiu anteriormente, é um processo relativamente simples.

Para a instalação dos três *clusters* pretende-se utilizar uma facilidade do Rocks que consiste em instalar *clusters* a partir de um servidor central via HTTP, um cenário aplicável ao desenvolvimento de grades computacionais usando o Rocks [27]. Para o caso deste trabalho segue-se a seguinte metodologia: 1) instalação do *cluster* C1, começando com a instalação e configuração do Rocks no nó frontal C1-n0 e depois

instalação dos nós de computação, C1-n1 e C1-n2, a partir do nó frontal; 2) instalação dos outros dois *clusters,* C2 e C3, a partir do nó frontal de C1, considerado para o efeito servidor central. A Figura 3.6 resume este procedimento.

Assim, a instalação de C2 consiste em: 1) instalação do seu nó frontal, C2-n0, a partir do nó frontal C1-n0 e de seguida, 2) instalação dos nós de computação com base em C2-n0. A instalação de C3 tem os mesmos procedimentos. Este tipo de instalação implica que os nós frontais a instalar de acordo com este método se encontrem registados em DNS.

Durante a instalação os *clusters* são configurados com os seguintes parâmetros:

- C1 – rede IP privada = 10.1.1.0, mascara = 255.255.255.0, *gateway* = 10.1.1.1, endereço público de nó frontal (C1-n0) = 192.168.10.1;
- C2 – rede IP privada = 10.2.2.0, mascara = 255.255.255.0, *gateway* = 10.2.2.1, endereço público de nó frontal (C2-n0) = 192.168.10.2;
- C3 – rede IP privada = 10.3.3.0, mascara = 255.255.255.0, *gateway* = 10.3.3.1, endereço público de nó frontal (C3-n0) = 192.168.10.3;
- A rede IP 192.168.10.0/24 simula uma rede pública que interliga os três *clusters*.

3.3 Rede privada virtual

Nesta secção apresenta-se uma outra tecnologia que é fundamental para o presente trabalho: rede privada virtual, mais conhecida como VPN (*Virtual Private Network*). Primeiro será feita uma breve introdução à tecnologia VPN [28] e a sua evolução integrando o protocolo SSL/TLS. De seguida é caracterizada a solução OpenVPN e o contexto de utilização neste trabalho.

3.3.1 Introdução

As VPNs estabelecem túneis IP entre dois ou mais sistemas informáticos (por ex.: computadores e encaminhadores) abstraindo as camadas de comunicações que entre eles existem. Este nível de abstracção equivale a estabelecer ligações directas entre os sistemas informáticos, como se de ligações físicas se tratassem.

Resumidamente, a técnica subjacente a esta tecnologia consiste em intersectar pacotes IP no seu percurso e encapsulá-los em novos pacotes que assim são transportados desde um sistema até a um outro sistema ligado numa rede diferente. Aqui estes pacotes são desencapsuládos dando lugar aos pacotes originais que, sendo agora colocados numa outra rede, prosseguem até ao seu destino.

Um cenário de utilização frequente consiste na ligação remota de um utilizador a uma rede privada da sua instituição. Por razões de segurança não se pretende que esta rede tenha comunicações directas com redes exteriores. A solução passa por estabelecer uma VPN entre o utilizador e a rede da instituição, ou melhor com um sistema de ligações VPN, integrado na rede. Neste caso os pacotes que saem do computador do

utilizador com destino à rede da instituição são encapsulados em novos pacotes com o endereço externo do sistema de ligações VPN. Este recebe os pacotes, desencapsula-os e coloca-os na rede interna da instituição para atingirem o seu destino.

Associadas ao encapsulamento estão outras duas funções que caracterizam uma VPN: autenticação e encriptação. A primeira valida a origem dos pacotes e a segunda cifra os dados contidos nos pacotes. O conjunto de funcionalidades assim integradas numa solução VPN, garantem a simulação de uma rede privada com as características que lhe são intrínsecas: confidencialidade e autenticidade.

Uma das implementações de VPN é baseada no protocolo IPSec (*IP Security*) [28], o qual introduz uma mudança complexa na própria pilha IP. IPSec examina pacotes que saem de uma interface IP, determina se existe alguma associação de segurança com o destino, e tenta automaticamente cifrar os pacotes na saída. Procede de forma inversa na chegada. Por se encontrar integrado na pilha protocolar IP é bastante eficiente.

Tem no entanto desvantagens das quais importa realçar uma que é importante no contexto deste trabalho. A sua instalação em sistemas Linux implica a recompilação do sistema operativo. Ora um dos propósitos definidos inicialmente consiste em integrar soluções que não exijam alterações substanciais nos sistemas operativos dos *clusters*.

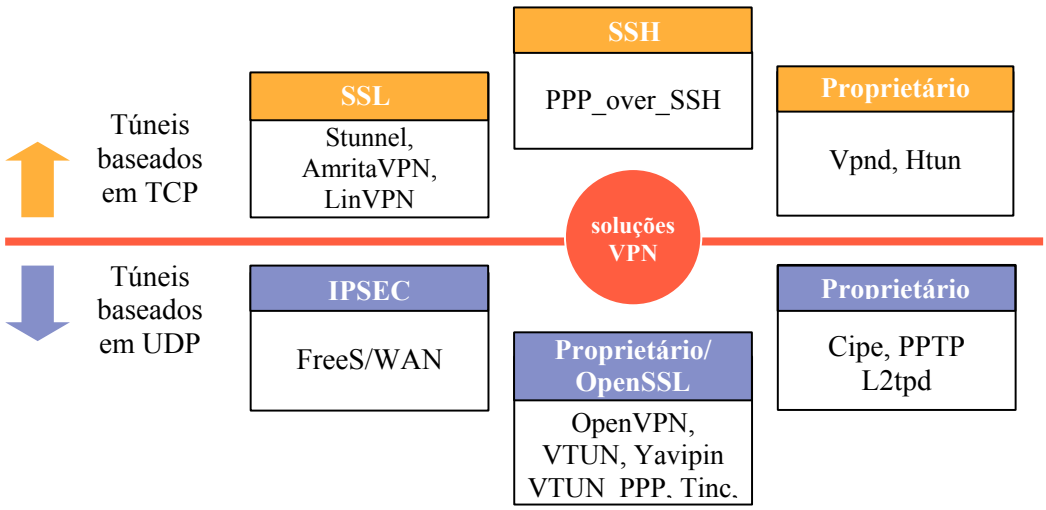

Figura 3.7 – Soluções VPN Espaço-Utilizador *versus* protocolos de segurança [32].

Uma alternativa às VPNs baseadas no IPSec é as VPNs que assentam na utilização do protocolo SSL (*Secure Sockets Layer*) adoptado pelo IETF como TLS (*Transport Layer Security*). Desde logo utilizam código que funciona em espaço-utilizador, não requerendo assim qualquer alteração no sistema operativo. Utilizam ainda uma interface de rede virtual que é visto pelo sistema operativo como uma interface do tipo *Point-to-Point*, designada por TUN (*Tunneling Network*). Esta interface recebe pacotes, reencaminha-os para um processo que procede à sua encriptação e encapsulamento numa ligação UDP e posteriormente envia-os para um sistema remoto que os desencripta, os autentica e finalmente os envia para a interface TUN do sistema remoto.

Sucintamente, este é o funcionamento do modelo VPN Espaço-Utilizador e demonstra a sua independência em relação ao *kernel* do sistema operativo, tornando-o um modelo mais flexível tanto na portabilidade para diferentes sistemas operativos (Windows, Linux, MacOS, etc.), como na instalação e manutenção.

Existem várias soluções deste tipo e de código aberto para sistemas Linux, algumas das quais utilizam como protocolo de segurança SSL/TLS ou usam a biblioteca OpenSSL, conforme se mostra na Figura 3.7. Pelas suas características, algumas destas soluções correspondem no geral aos requisitos do trabalho já referidos. No entanto destacam-se duas: Tinc [30] e OpenVPN [31]. Em artigo do *IEEE Communications Magazine* [32] é feito uma análise global abordando as principais questões relacionadas com funcionamento de VPNs: desempenho de rede, nas vertentes de sobrecarga (*overhead*), largura de banda e latência, funcionalidades suportadas, relativamente a encaminhamento IP e modularidade do código, aspectos operacionais, como segurança e escalabilidade, e ainda complexidade de configuração.

De acordo com o estudo analisado e como já foi referido, tanto o OpenVPN como o Tinc constituem soluções interessantes e assim foi considerada a sua integração na rede de interligação dos *clusters*. Pelas razões referidas mais à frente, no Capítulo 4, considera-se o OpenVPN globalmente mais adequada para arquitectura de rede que se pretende implementar.

De seguida é caracterizada de forma sumária a solução OpenVPN.

3.3.2 OpenVPN

OpenVPN é uma solução VPN Espaço-Utilizador desenvolvida sob GPL da GNU que utiliza o protocolo SSL/TLS para garantir as mesmas funcionalidades de segurança implementadas por soluções baseadas em IPSec ou PPTP (*Point-to-Point Tunneling Protocol*). Utiliza o mesmo programa em ambos os lados de uma ligação, qualquer que seja o modo de funcionamento.

Suporta os modos de funcionamento ponte e encaminhador. No primeiro, os túneis criados sobre interfaces TAP – interface de rede Ethernet virtual – permitem a transmissão de tráfego Ethernet. No segundo modo, os túneis implementados com interfaces TUN – interface de rede Point-to-Point virtual – encaminham tráfego entre redes IP. Utiliza preferencialmente o protocolo UDP, mas também TCP, para transmitir todo o tipo de tráfego sobre um único porto, normalmente o 1194.

Tratando-se de uma aplicação que corre no espaço-utilizador o seu desempenho depende da plataforma de hardware que o suporta. Por outro lado, os processos OpenVPN não têm um impacto significativo na carga de processamento dos CPUs. Assim, o OpenVPN não tem um limite para o número de túneis que pode estabelecer.

Permitindo mais que um túnel entre dois sistemas, OpenVPN possibilita a implementação de cenários de partilha de carga, sendo a tarefa de distribuição de tráfego da responsabilidade de um processo externo, por exemplo o IPtables - pacote de software para controlo de tráfego.

Ao contrário do IPSec, o OpenVPN possibilita o estabelecimento de túneis atravessando sistemas que implementem NAT, uma vez que o TLS não implementa autenticação com base no endereço IP origem.

Em termos de autenticação inicial o OpenVPN pode usar o método de chaves partilhadas (modo *Static Key*), que constitui sérios riscos de segurança, ou em alternativa recorrer a autenticação baseada em certificados digitais X509, suportada integralmente por PKI (Public *Key Infraestruture*) do OpenSSL (modo SSL/TLS).

O modo TLS é o modo criptográfico mais poderoso de OpenVPN em termos de segurança e flexibilidade. Funciona estabelecendo canais de controlo e canais de dados, os quais são multiplexados sobre um único porto TCP/UDP. OpenVPN inicia uma sessão TLS sobre um canal de controlo e utiliza-o para trocar as chaves de encriptação e de HMAC. TLS utiliza uma robusta camada de segurança sobre a ligação UDP para toda a comunicação do canal de controlo, enquanto o canal de dados, sobre o qual se estabelecem túneis de dados cifrados, é encaminhado sem qualquer mediação. O resultado é o melhor de dois mundos: um canal de dados rápido que transmite sobre UDP somente com a sobrecarga da encriptação/desencriptação, e funções HMAC, e um canal de controlo que fornece todas as funcionalidades de segurança do TLS, incluindo autenticação baseada em certificados digitais e chave *Diffie Hellman* [33].

Para usar o modo TLS, cada sistema que executa o OpenVPN deve ter localmente o seu par certificado/chave privada assinado pelo certificado da CA.

Quando dois sistemas OpenVPN se ligam, cada um apresenta o seu certificado ao outro. Cada sistema verifica então se o seu parceiro apresentou um certificado assinado pelo certificado da CA. Se a verificação em ambos os sistemas é bem sucedida, então a negociação TLS será concluída com êxito, ambos trocarão chaves temporárias de sessão, e o túnel iniciará a transmissão de dados. A distribuição OpenVPN contém um conjunto de utilitários para gerir certificados e chaves RSA.

A instalação do OpenVPN é em geral muito simples, sendo mesmo uma das suas vantagens. Integra já algumas distribuições de Linux. Para sistemas Red Hat, utilizados nos *clusters* Rocks, são disponibilizadas versões em RPM. Como requisito especial exige a instalação das interfaces virtuais TUN e TAP que se encontram instalados em sistemas usando Linux Kernel 2.4.x ou superior. A compilação pode ser feita com opção de compressão usando a biblioteca LZO.

O OpenVPN tem um grande número de opções. Na configuração devem ter-se algumas precauções e incluir algumas directivas importantes. É conveniente diminuir os privilégios do processo `openvpn` depois de executado, usando as directivas `user`/`root` `nobody` e `chroot`. Para aumentar a segurança pode ser activada a opção `auth-tls` para utilizar um nível de autenticação baseado em HMAC. Também é importante uniformizar o MTU para evitar a fragmentação de pacotes e a consequente degradação de desempenho, introduzindo as opções `tun-mtu 1500` e `mss-fix 1400`.

3.3.3 Contexto de utilização

OpenVPN integra a solução projectada implementando a rede de interligação de *clusters*, a qual consiste em ligações VPN ponto-multiponto estabelecidas por cada nó frontal com todos os outros nós frontais, conforme se mostra na Figura 3.8. Cada *cluster*

é constituído por um conjunto de nós de computação interligados por uma rede IP privada, e por um nó frontal que estabelece a ligação desta rede privada com o exterior.

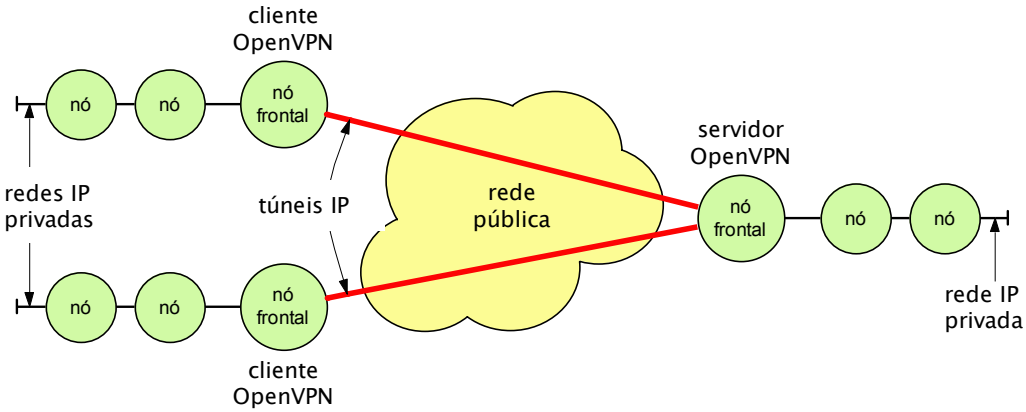

Figura 3.8 – Cenário de utilização do OpenVPN.

Neste cenário o OpenVPN é utilizado para interligar as redes IP privadas dos *clusters*, estabelecendo uma malha de túneis de cada rede com todas as outras. Sendo assim configurado no modo de encaminhador e TLS. Isto é, num cenário com N Clusters, cada servidor (um nó frontal) estabelece N-1 túneis IP e é usado o protocolo TLS para autenticar e cifrar cada túnel.

A utilização do modo TLS implica a criação de uma Autoridade de Certificação (CA) para a emissão de certificados digitais e chaves privadas para servidores e clientes OpenVPN, bem como, a chave *Diffie Hellman* (DH). Para isso será implementada uma PKI OpenSSL num dos nós frontais, a partir do qual será a feita a distribuição de certificados e chaves.

Na instalação são feitas algumas opções. A compressão de dados não é activada, por se considerar que é um factor que incrementa a latência na comunicação, já considerável neste tipo de VPNs Espaço-Utilizador. É utilizado o protocolo UDP configurado no porto 1194. Não é utilizada a autenticação HMAC, por implicar uma sobrecarga administrativa, uma vez que é necessário partilhar entre todos os nós uma chave HMAC para ser usada no processo de negociação do TLS. É uniformizado o MTU entre todos os nós. São usadas as opções de encaminhamento disponíveis no OpenVPN para actualizar as tabelas de encaminhamento IP dos nós, assim que se estabelecem as ligações.

3.4 Procurador de NFS

Importa agora abordar um outro componente que é utilizado neste trabalho: procurador NFS. Depois de uma breve descrição do NFS é explanado o funcionamento de um procurador de NFS no sentido de compatibilizar sistemas NFS de domínios de administração diferentes. Posteriormente, é apresentado o contexto de utilização dos procuradores NFS na solução projectada.

3.4.1 Introdução

Aflorando o serviço NFS (*Network File System*) [34] pode afirmar-se que se trata de um sistema de ficheiros distribuído que disponibiliza acesso transparente a discos remotos. Funciona no modelo cliente-servidor, em que o servidor publica parte do seu sistema de ficheiros que posteriormente o cliente integra no seu sistema de ficheiros local. Esta integração possibilita ao cliente a utilização indiferenciada das duas componentes do seu sistema de ficheiros: a local e a remota sedeada no servidor.

A transparência do NFS traduz-se na capacidade de execução das mesmas operações, quer os ficheiros sejam locais ou remotos, por parte dos utilizadores e aplicações do cliente, mas também no facto das mesmas operações poderem ser realizadas de igual forma, quer do lado do cliente quer do lado do servidor. Há portanto uniformidade, coerência, na representação dos ficheiros neste sistema distribuído. Um ficheiro conserva os seus atributos quer seja utilizado no servidor ou no cliente.

Uma parte da transparência é habitualmente assegurada pelo serviço NIS (*Network Information System* [34] encarregue de divulgar por todos os elementos do sistema distribuído informação sobre a sua configuração, incluindo utilizadores. A coerência da informação ao longo de todos os elementos repercute-se no sistema de ficheiros que só é possível porque todos os elementos integram o mesmo domínio de administração.

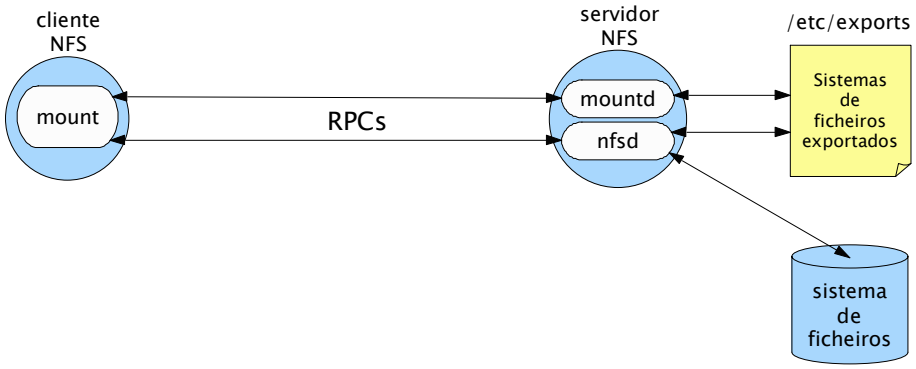

Figura 3.9 – Processos do serviço NFS.

Da arquitectura do NFS importa realçar que utiliza RPCs (*Remote Procedure Call*) e que contempla um processo cliente, `mount`, e um processo servidor, `nfsd`. Um outro processo, `mountd`, embora não integre o NFS, também intervém no seu funcionamento. O processo `nfsd` recebe pedidos RCP dos clientes solicitando acesso aos ficheiros no servidor. O acesso é executado pelo `nfsd` de acordo com políticas definidas. Já o processo `mountd` recebe pedidos dos clientes solicitando montar sistemas de ficheiros do servidor. Por sua vez, o servidor NFS utiliza `mountd` para informar os clientes quais os sistemas de ficheiros disponíveis (exportados) para montar. Os sistemas de ficheiros exportados e as respectivas políticas de acesso são definidos no ficheiro `exports`. A Figura 3.9 resume de forma simplificada os processos NFS, cuja comunicação pode ser suportada pelos protocolos TCP ou UDP.

Deve-se ainda referir que o NFS implementa sistemas de *cache* tanto do lado servidor como do cliente, podendo opcionalmente ser activadas.

3.4.2 Procurador PUNCH VFS

Feita esta síntese do serviço NFS é altura de abordar o componente Procurador NFS integrado na solução.

Como se referiu em cima, para que o serviço NFS funcione é fundamental a partilha de um sistema de informação que mantenha os sistemas de ficheiros partilhados consistentes entre todos os elementos que integram um domínio NFS. No caso de um elemento não possuir a sua informação sincronizada com o servidor NFS, pode acontecer que os seus utilizadores não possam aceder a ficheiros partilhados pelo servidor, pelo simples facto de os atributos de utilizadores e atributos de ficheiros não coincidirem, nomeadamente os atributos UID/GID.

Isto é o que normalmente acontece, por exemplo, entre sistemas NFS integrados em domínios de administração diferentes, não havendo forma directa de uniformizar informação entre os diferentes domínios.

Um Procurador NFS é um processo que recebe pedidos RCP de um cliente, ou de outro Procurador, os processa e os reenvia para o servidor NFS. E envia as respostas no sentido inverso. O processamento corresponde a autenticar e autorizar os pedidos e pode modificar os atributos dos ficheiros envolvidos nos pedidos, tendo por isso uma política de acessos associada.

A solução de Procurador NFS que é utilizada no trabalho foi desenvolvida pelo Laboratório de Computação Avançada e Sistemas de Informação (ACIS) da Universidade da Califórnia, nos Estados Unidos, no âmbito da grade computacional PUNCH, já referida no Capítulo 2. Concretamente é a base do PUNCH *Virtual File System* (PVFS) [35], um sistema de ficheiros virtual que permite a transferência dinâmica de dados entre unidades de armazenamento e sistemas servidores durante uma sessão de computação na grade PUNCH. Tem a particularidade de não implicar modificações em aplicações já desenvolvidas para sistemas convencionais. PVFS emprega procuradores NFS para estabelecer transacções entre clientes e servidores NFS normais, sendo os procuradores dinamicamente configurados e controlados por software da grade computacional.

O Procurador PVFS é composto por dois processos: `pvfs.mountd` e `pvfs.nfsd`. O primeiro autentica pedidos de um cliente NFS ou de outro procurador, solicitando utilizar directorias remotas. Os pedidos são autorizados de acordo a configuração definida no ficheiro `mountd_exports`. Caso os pedidos sejam autenticados são reenviados para um servidor de NFS ou para outro procurador. O segundo processo autentica pedidos de acesso a directorias exportadas de acordo com a política definida nos ficheiros `nfsd_exports` e `map`. Os pedidos validados são igualmente reenviados para um servidor NFS ou para outro procurador.

Como se mostra na Figura 3.10, enquanto em `mountd_exports` se definem os sistemas de ficheiros partilhados e seus utilizadores, em `nfsd_exports` define-se para cada cliente, ou grupo de clientes, o tipo de acesso (leitura/escrita) permitido. Em ambos os ficheiros de configuração podem ser definidas políticas que são subconjuntos das

políticas definidas a montante, no servidor NFS ou em outro procurador. Estes dois ficheiros têm um formato semelhante ao `/etc/exports` do NFS:

- mountd_exports

  directoria exportada → clientes autorizados a usá-la

- nfsd_exports

  directoria exportada → clientes autorizados (escrita/leitura, mapeamento identidade)

O mapeamento de identidade referido em cima é definido no ficheiro `map`, onde se indica qual a conversão da identidade (UID/GID) entre o utilizador proprietário da directoria exportada pelo servidor e a identidade do utilizador no cliente onde a directoria é montada. Na prática `map` indica qual o mapeamento entre os atributos UID/GID da directoria no servidor e no cliente NFS. Tem o formato:

UID (no cliente → no servidor)
GID (no cliente → no servidor)

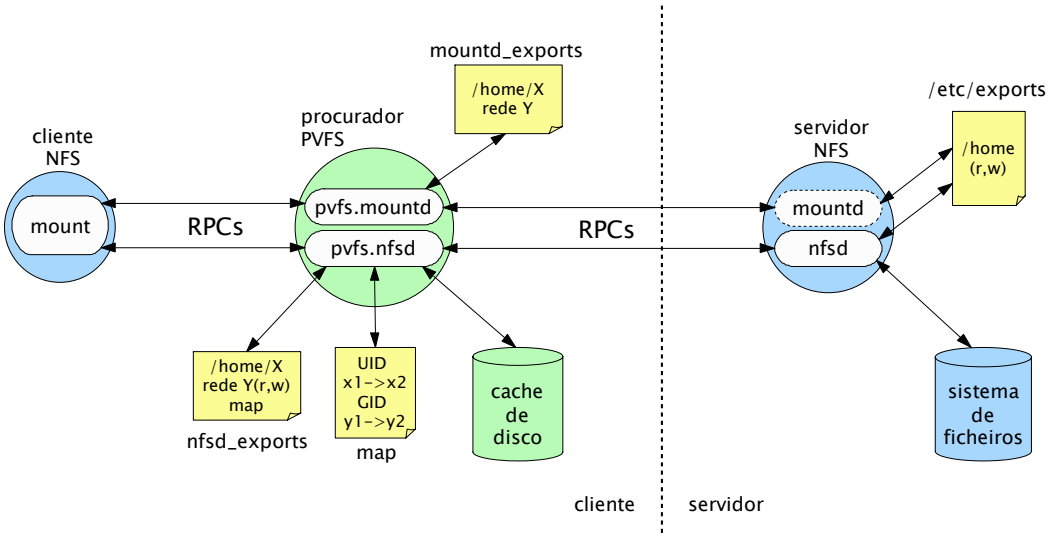

Figura 3.10 – Modelo de funcionamento de Procurador PVFS

Uma funcionalidade importante do Procurador PVFS é o seu sistema de *cache* em disco [36] que permite melhorar o desempenho de sistema de ficheiros distribuído implementado em NFS, uma vez que dados lidos do servidor são entregues ao cliente e gravados na *cache* de disco. As próximas leituras dos mesmos dados são feitas da *cache* e não do servidor. A implementação desta *cache* tem a particularidade de suportar configurações ajustadas quer a utilizadores quer a aplicações.

A *cache* é organizada em bancos de ficheiros formando uma estrutura que armazenam blocos de dados e etiquetas da *cache*. Os bancos da *cache* são criados no disco local do sistema que suporta o procurador. Para as operações de escrita suporta

diferentes políticas: só de leitura, escrita directa, e escrita para trás. Esta última é importante em cenários caracterizados por comunicações com grande latência, como é caso das redes alargadas. A configuração da *cache* deve ter em conta a integridade e a consistência dos dados e para isso são também implementados mecanismos de sincronismo entre a *cache* e o servidor de ficheiros [37].

Conforme é representado na Figura 3.10, o procurador deve funcionar do lado do cliente, embora também possa coexistir procurador do lado do servidor, formando uma hierarquia de procuradores.

3.4.3 Contexto de utilização

Os Procuradores PVFS são utilizados na implementação de um sistema de ficheiros da plataforma *multi-cluster*, descrito no Capítulo 4. Com a mesma funcionalidade mas num contexto mais restrito, volta a ser referenciado no Serviço de *Cluster* Virtual, explanado no Capítulo 5. No âmbito destas soluções a designação utilizada é Procurador NFS.

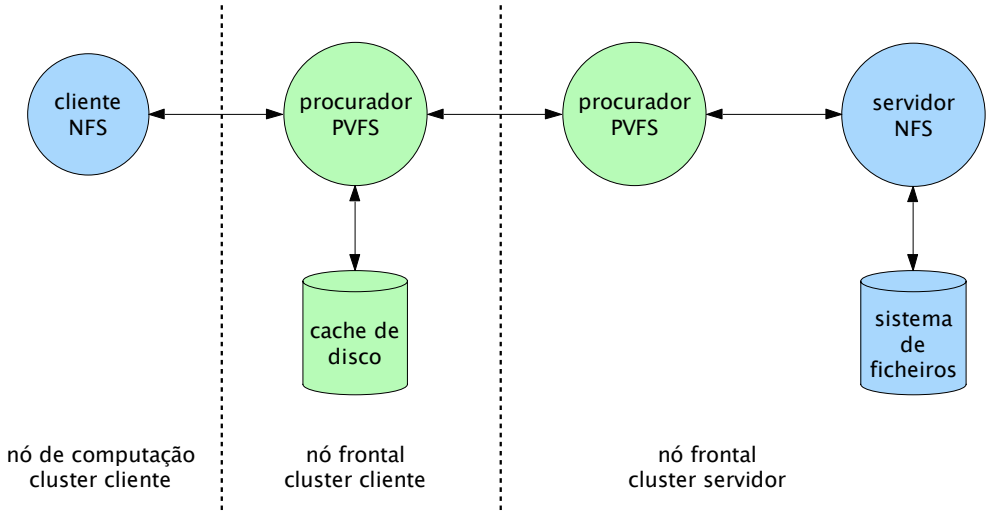

Figura 3.11 – Hierarquia de procuradores PVFS utilizada na solução.

Resumidamente, a utilização consiste na implementação de uma hierarquia de dois procuradores. Um que funciona junto do servidor NFS nativo e outro que funciona junto do cliente. Geograficamente, tanto podem estar separados em nós frontais de *clusters* diferentes, como podem correr no mesmo nó frontal. O primeiro cenário é o mais frequente. Como se ilustra na Figura 3.11 a hierarquia NFS completa é composta por:

- Cliente NFS habitualmente correndo num nó de computação, também pode correr num nó frontal;
- Procurador PVFS instalado no nó frontal do mesmo *cluster* onde está o cliente NFS, também pode funcionar no nó frontal do *cluster* servidor
- Procurador PVFS que funciona sempre no nó frontal do *cluster* onde corre o servidor NFS;

- Servidor NFS nativo de um *cluster* a funcionar no seu respectivo nó frontal;
- Por último o sistema de ficheiros partilhado por NFS.

O procurador junto do servidor corre permanentemente, é activado juntamente com os serviços NFS, enquanto que o procurador do cliente só executa durante a sessão de um utilizador. A Figura 3.11 também esclarece que é utilizada *cache* de disco no procurador instalado do lado do cliente NFS, a qual é configurada com a política de "escrita para trás" de forma a diminuir o número de operações de escrita no servidor NFS. É ainda activado o sincronismo entre a *cache* e o servidor para que se garanta a coerência e a integridade dos dados, certamente partilhados com outros clientes NFS.

Foram assim descritas as principais tecnologias utilizadas no presente trabalho. Nos próximos capítulos expõe-se a solução que integra estas tecnologias.

Capítulo 4

Plataforma *multi-cluster*

Partindo de um conjunto de recursos constituído por diversos *clusters* computacionais geograficamente distribuídos e inseridos em domínios de administração diferentes, pretende-se formar uma plataforma computacional alargada, de modo a proporcionar uma utilização mais eficiente e flexível, abrangendo mais utilizadores e projectos de maior amplitude requerendo bastos recursos, em quantidade, qualidade e diversidade.

É importante referir que a constituição desta nova plataforma deve considerar os *clusters* a integrar como recursos consolidados, com utilizadores e aplicações estabilizadas e que portanto as novas componentes e respectivos constituintes devem provocar o menor impacto possível nos vários níveis dos *clusters* existentes: arquitectura, administração, ambientes de programação e execução. Este é um aspecto fundamental do projecto.

4.1 Introdução

Como se referiu em cima, esta nova plataforma computacional será baseada em *clusters*, os quais partilham características que se consideram essenciais, ou mesmo requisitos: arquitectura *Beowulf*, sistema de operativo Linux e sistema de gestão NPACI Rocks, apresentado no capítulo 3.

É portanto comum a estes *clusters* uma arquitectura que assenta em dois tipos de nós, nó frontal e nó de computação, e numa rede privada de comunicação, com bom desempenho. No modelo de *multi-cluster* projectado, considera-se o *cluster* tipo constituído por um nó frontal e vários nós de computação interligados por uma rede Ethernet (Fast ou Gigabit Ethernet), suportando TCP/IP, e com endereçamento privado. Os nós de computação executam aplicações paralelas enquanto o nó frontal, com ligação também a uma rede pública, garante a interacção com o utilizador – possibilitando a este submeter e monitorizar trabalhos, introduzir dados e receber resultados – e ainda fornece vários serviços ao *cluster*.

Para além dos serviços relacionados com a administração do *cluster*, oferecidos pelo Rocks, interessa destacar outros serviços que terão relevância na solução de plataforma *multi-cluster* que se apresenta neste capítulo. Desde logo o serviço de armazenamento de ficheiros centralizado no nó frontal usando NFS, permite aos utilizadores partilharem as suas aplicações e dados entre todos os nós de computação. Os serviços de encaminhamento de tráfego e NAT (*Network Address Translation*) facilitam aos nós de computação a comunicação com o exterior. Por último, o serviço de

controlo de tráfego (*firewall*) garante a segurança de todos os nós do *cluster*. A Figura 4.1 ilustra este modelo de *cluster* utilizado na plataforma *multi-cluster*.

Nesta arquitectura os nós de computação comunicam directamente entre si através da rede IP privada. A comunicação com o exterior é assegurada pelo nó frontal mas só no sentido de dentro para fora, isto é, apenas para comunicações com origem nos nós de computação. A comunicação directa com origem em máquinas externas ao *cluster* e destinada aos nós de computação não é viável.

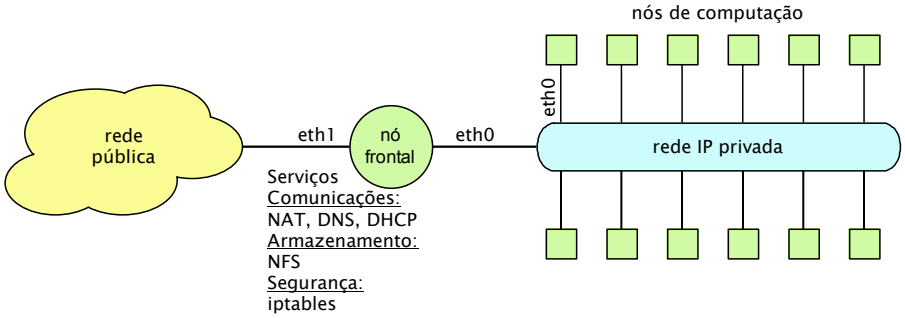

Figura 4.1 – Arquitectura de *cluster* participante na plataforma *multi-cluster*.

4.2 Interligação de *clusters*

A plataforma *multi-cluster* será constituída por *clusters* implementando esta arquitectura. Pretende-se que aplicações paralelas possam correr num ambiente de execução alargado aos nós do *multi-cluster*. Ou seja, é necessário assegurar a comunicação entre todos os nós de computação, apesar de se encontrarem ligados às redes privadas dos respectivos *clusters*.

É sabido que as redes IP privadas não podem comunicar entre si usando infra-estruturas de comunicações públicas, pela simples razão de não existirem rotas para encaminhamento de tráfego que eventualmente se lhes destina. Assim sendo, surge desde já um problema que o projecto precisa de resolver: interligar as redes IP privadas dos *clusters* que compõem a plataforma *multi-cluster*. Isto é, permitir que um nó do *cluster* A consiga comunicar directamente com outro nó do *cluster* B (ver Figura 4.2).

Solução VPN

As redes privadas dos *clusters* utilizam redes IP inválidas. Normalmente, o Rocks utiliza a rede de classe A 10.0.0.0/8. A interligação destas redes inválidas com redes públicas (redes IP válidas) é assegurada pelos nós frontais, permitindo assim que os utilizadores possam interagir com os *clusters* a partir de qualquer ponto da Internet.

Em ambiente *multi-cluster* é necessário, porém, que os nós de diferentes *clusters* comuniquem entre si. Para isso deve estabelecer-se uma rede de interligação dos diversos *clusters* que compõem o *multi-cluster*. Por outras palavras, é desejável construir uma WAN (*Wide Área Network*) com base em redes públicas que interligue

todos os *clusters*, permitindo a comunicação directa entre dois quaisquer nós de computação do *multi-cluster*.

Este requisito levanta alguns problemas. Por um lado, uma WAN, por si só, não resolve o problema da comunicação entre nós de diferentes *clusters*. Tratando-se de redes IP inválidas interligadas através de redes públicas, significa que, nestas últimas, não existem rotas de encaminhamento de tráfego que permitam aos pacotes que saem da rede privada de um *cluster* chegarem à rede privada de outro *cluster*. Por outro lado, a utilização de redes públicas implica certamente que o tráfego atravesse *firewalls* e outros sistemas de protecção que podem impedir a comunicação entre *clusters*. Por outro lado ainda, são também relevantes os problemas de segurança da informação que será trocada entre *clusters* sobre redes públicas de comunicações.

Estes problemas serão ultrapassados utilizando, sobre a rede WAN de interligação, a tecnologia VPN (*Virtual Private Network*). A solução proposta consiste em implementar ligações VPN entre os nós frontais dos *clusters*. Desta forma, estabelecem-se túneis de comunicação entre as redes IP privadas que garantem não só a conectividade entre elas, como também a confidencialidade da informação que é transmitida.

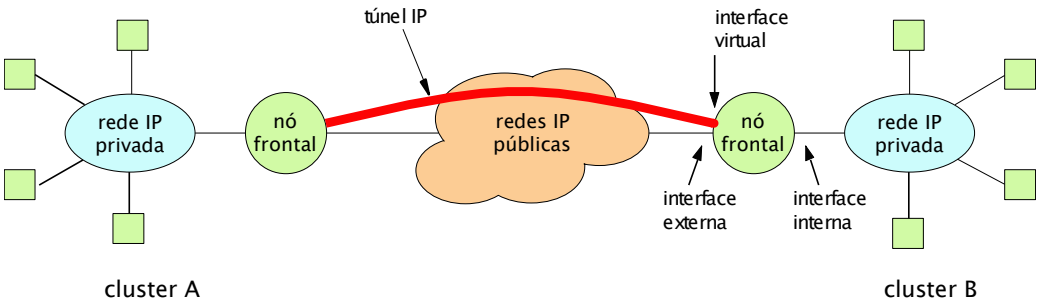

Figura 4.2 – A VPN entre dois *clusters* estabelece um túnel IP entre os nós frontais.

Em termos práticos o estabelecimento do túnel IP entre dois *clusters*, representado na Figura 4.2, introduz um nível de abstracção que "elimina" todas as ligações físicas que na realidade existem entre os dois *clusters*, ligando virtualmente os dois nós frontais directamente entre si. Permitindo, assim, que as tabelas de encaminhamento destes dois nós passem a contemplar rotas para as redes privadas dos *clusters*. Como resultado desta alteração o tráfego com origem na rede privada do *cluster* A pode chegar à rede privada do *cluster* B e vice-versa.

Tabela 4.1 – Rotas de encaminhamentos nos nós frontais antes da ligação VPN.

Cluster A – Nó Frontal		
Rota	**Rede IP**	**Interface**
1	Privada de A	Interna
2	Outras redes	Externa

Cluster B – Nó Frontal		
Rota	**Rede IP**	**Interface**
1	Privada de B	Interna
2	Outras redes	Externa

Antes da ligação VPN entre os dois *clusters* as tabelas de encaminhamento IP dos respectivos nós frontais podem ser resumidas conforme se apresentam na Tabela 4.1.

Depois da ligação VPN cada nó frontal passa a ter uma interface virtual para a qual é encaminhado o tráfego com destino à rede privada do outro *cluster*. A Tabela 4.2 reflecte essa alteração nas tabelas de encaminhamento.

Tabela 4.2 – Rotas de encaminhamentos nos nós frontais depois da ligação VPN.

Cluster A – Nó Frontal		
Rota	**Rede IP**	**Interface**
1	Privada de A	Interna
2	Privada de B	Virtual (VPN)
3	Outras redes	Externa

Cluster B – Nó Frontal		
Rota	**Rede IP**	**Interface**
1	Privada de B	Interna
2	Privada de A	Virtual (VPN)
3	Outras redes	Externa

A segurança informática é outro aspecto que valoriza esta solução VPN. De facto este tipo de solução não coloca em causa a segurança inerente à rede privada de cada *cluster*. Por um lado, continua a não ser possível qualquer máquina no exterior estabelecer conexões com os nós de computação, apenas é permitida a comunicação entre os nós dos dois *clusters*. Por outro lado, a informação que circulará entre os dois *clusters* será cifrada e autenticada.

Para assegurar a ligação VPN eventualmente será necessário alterar algumas regras de segurança do nó frontal, nomeadamente, autorizar a comunicação através de um determinado porto UDP/TCP e da interface virtual.

Como se ilustra na Figura 4.2 a implementação de uma ligação VPN entre os nós frontais de dois *clusters* é relativamente simples, mas quando aumenta o número de *clusters* é necessário planear um modelo capaz de garantir fiabilidade e escalabilidade.

A solução final pode consistir: 1) na definição de uma malha de túneis ponto-a-ponto entre todos os nós frontais do *multi-cluster*; 2) ou numa implementação que se baseie em VPNs ponto-multiponto.

Tendo como princípio a utilização de soluções que não impliquem modificações substanciais nos sistemas operativos dos nós frontais, enveredou-se por VPNs de código aberto baseadas em SSL/TLS como alternativa às soluções que utilizam IPSec. Assim, identificaram-se dois pacotes de software: Tinc e OpenVPN. Duas propostas que sugerem arquitecturas diferentes para a rede de interligação, conforme é referido no parágrafo anterior.

4.2.1 Rede com ligações ponto-a-ponto

A solução Tinc permitiria uma rede de interligação de N *clusters* como uma sequência de ligações VPN entre os nós frontais dos *clusters*, conforme se pode constatar na Figura 4.3.

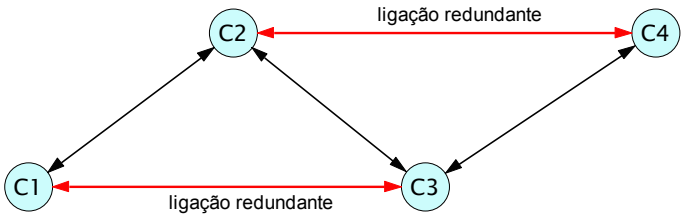

Figura 4.3 – Grafo das ligações VPN entre 4 *clusters* usando a solução Tinc.

Esta solução tem a vantagem de implicar um número reduzido de ligações VPN. Para interligar N *clusters* seriam necessárias N-1 ligações VPN. Também se poderá deduzir que em termos de gestão seria uma solução simples. No entanto, apresenta duas principais desvantagens.

A primeira é que cada nó frontal funciona como um encaminhador de tráfego IP destinado a outros clustes. Por exemplo, o nó frontal do *cluster* 3 encaminha todo o tráfego com origem nos *clusters* C1 e C2 destinado aos restantes *clusters* e vice-versa. Este modelo implica uma maior, e indesejável, sobrecarga de processamento de cada nó frontal, ou visto noutra perpectiva, um mau desempenho de um nó frontal corresponde a uma maior latência na comunicação.

A segunda desvantagem resulta de que uma eventual quebra numa ligação VPN comprometeria a comunicação entre os *clusters* de ambos os lados da quebra. Esta desvantagem pode ser resolvida estabelecendo ligações redundantes, conforme se apresenta na Figura 4.3, de modo a que entre dois quaisquer *clusters* haja sempre, pelo menos, dois caminhos alternativos. Consequentemente, esta implementação exige a gestão de rotas de encaminhamento de tráfego IP, em cada nó frontal, os quais neste contexto funcionam como encaminhadores IP.

O software Tinc integra módulos que fazem a gestão dinâmica de rotas de encaminhamento IP, aliás este é um dos seus pontos fortes. Num cenário de quebra de uma ligação, as tabelas de encaminhamento IP nos nós frontais são ajustadas automaticamente para que se utilizem as ligações alternativas.

Tinc constitui assim uma solução muito interessante, fácil de implementar, garantindo com grande transparência a conectividade entre todos os *clusters*. No entanto, a fiabilidade da rede de interligação fica sempre dependente do bom desempenho de todos os nós frontais. Uma sobrecarga de processamento provocada por aplicações locais num determinado nó pode afectar as comunicações entre nós, assim como um elevado volume tráfego pode afectar consideravelmente o desempenho de aplicações locais de um dado nó frontal.

4.2.2 Rede com ligações ponto-multiponto

Pelas razões apontadas na secção anterior, considerou-se a utilização de um modelo com topologia em estrela, em que cada *cluster* estabelece ligações com todos os outros *clusters*. Ou melhor, cada nó frontal de um *cluster* estabelece ligações VPN com todos os nós frontais dos outros *clusters*.

A solução poderá ter algumas limitações em termos de escalabilidade, essencialmente porque cada nó frontal estabelece N-1 ligações VPN numa plataforma com N *clusters*, implicando a utilização considerável de recursos dos nós frontais e uma sobrecarga administrativa. Apesar disso constitui um óptimo compromisso entre o desempenho da rede de interligação e a escalabilidade numa plataforma com algumas dezenas de *clusters*. Refira-se a importância de factores como sobrecarga, largura de banda e latência no desempenho de uma VPN e as suas consequências num ambiente alargado de execução de aplicações paralelas. Neste sentido, esta solução apresenta vantagens nomeadamente em relação à latência. Uma vez que cada nó frontal estabelece ligações directas com os outros nós frontais, não havendo na ligação VPN entre dois nós outros nós intermédios que introduzam atrasos na comunicação, como acontece na solução Tinc.

Serve de base a este modelo o software de código aberto OpenVPN já apresentado no capítulo 3 dedicado às tecnologias. Um dos modos de funcionamento do OpenVPN é em cliente-servidor. Um único processo `openvpn` a correr como servidor recebe pedidos de ligação de clientes que depois de autenticados são aceites e se traduzem no estabelecimento de túneis IP cifrados. Uma solução OpenVPN deste tipo requer criação de uma estrutura de suporte à segurança, comum às soluções VPN. Concretamente, a solução OpenVPN em modo cliente-servidor requer:

- A criação de uma Autoridade de Certificação (CA) para a emissão de certificados X.509;
- A emissão de chaves privadas para servidor e clientes;
- A emissão de certificados X.509 para servidor e clientes assinados por CA;
- A emissão de uma chave DH (*Diffie-Hellman*) por servidor;
- A atribuição de rede IP (privada) para interligação.

A arquitectura preconizada para a interligação de N *clusters* assenta em dois princípios: 1) cada nó frontal é simultaneamente servidor e cliente VPN, com a excepção do nó de C_1 que é só cliente e do nó frontal de C_n que é somente servidor e 2) o nó frontal do *cluster* C_k que integra a plataforma *multi-cluster* constitui-se como servidor, ao qual se ligam como clientes todos os nós frontais dos outros *clusters* já existentes, C_1 a C_{k-1} (em que K é qualquer inteiro entre 2 e N).

Estes dois princípios implicam que numa plataforma com N *clusters* se estabeleça uma rede de interligação formada por uma malha com um número ligações expresso pelo seguinte somatório:

$$\text{Total\_ligações } (N \textit{ clusters}) = \sum_{i=1}^{N} \text{i-1}$$

Considerando a Figura 4.4, a aplicação dos dois princípios acima enunciados resulta que:

- C2 servidor tem como cliente C1;
- C3 servidor tem como clientes C1 e C2;

- C4 servidor tem como clientes C1, C2 e C3;
- C1 é somente cliente enquanto C4 é apenas servidor.

A direcção das linhas representadas na Figura 4.4 indica que as ligações VPN têm origem no servidor e terminação no cliente. A figura também ilustra o facto de que um nó frontal estabelece N-1 ligações VPN, independentemente do modo, cliente ou servidor, em que são feitas. No caso deste exemplo todos os nós estabelecem 3 ligações VPN. O nó frontal de C1 estabelece todas as ligações como cliente, e por isso correrá 3 processos `openvpn`, enquanto o nó de C4 estabelece as 3 ligações como servidor e por isso apenas executará um processo `openvpn`.

O número de processos `openvpn` que um nó frontal poderá ter em execução será um dos factores limitativos à escalabilidade, essencialmente pelo consumo de memória principal. No entanto, tendo em conta as actuais configurações de hardware destes servidores, considera-se que para uma plataforma de média escala este factor terá um impacto reduzido.

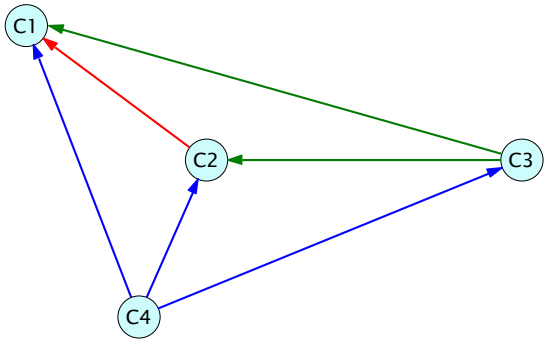

Figura 4.4 – Grafo da interligação de 4 *clusters* usando software OpenVPN.

Nesta arquitectura, apesar de se obter uma malha de ligações entre *clusters*, a comunicação entre dois *clusters* faz-se directamente, ponto-a-ponto, sem outros *clusters* intermediários. Cada nó frontal disporá de uma tabela de encaminhamento com rotas directas para as redes privadas dos restantes *clusters*. Exemplificando com o caso anterior, a tabela de encaminhamento do nó frontal do *cluster* C1 teria a forma abreviadamente apresentada na Tabela 4.3.

Tabela 4.3 – Exemplo de encaminhamento IP na arquitectura ponto-multiponto.

Rede IP destino	Encaminhador seguinte
Rede privada de C2	Nó frontal de C2
Rede privada de C3	Nó frontal de C3
Rede privada de C4	Nó frontal de C4

Não se colocam assim, as questões de encaminhamento de tráfego que foram abordadas na arquitectura de rede com ligações ponto-a-ponto, com o software Tinc. As actuais versões do OpenVPN permitem configurações que adicionam de forma automática estas rotas assim que as ligações VPN se estabelecem, bem como o inverso.

O OpenVPN caracteriza-se pela sua facilidade de configuração, mesmo no modo de funcionamento ponto-multiponto. No entanto, a relativa complexidade que o modelo pode atingir numa plataforma *multi-cluster* de muitos *clusters*, pode constituir outro factor impeditivo da escalabildade da solução. Assim, de forma minimizar a árdua tarefa de administrar um grande número de configurações, introduz-se agora o conceito de *Dominío OpenVPN* que visa organizar as configurações de cada nó frontal, de forma que cada configuração seja construída com base numa réplica da configuração anterior adicionando-lhe novos elementos.

Dominío OpenVPN

Numa solução OpenVPN que interligue N *clusters*, teremos, como já se viu, N-1 servidores VPN, cada um com os seus respectivos clientes. Isto significa que será necessário proceder-se à configuração de N-1 servidores e respectivos clientes.

Sendo cada configuração simples, é no entanto conveniente definir uma forma de planear o conjunto das configurações. Surge assim no âmbito da plataforma *multi-cluster*, a abordagem do conceito *Domínio OpenVPN*, segundo o qual a configuração global é decomposta em configurações parcelares. Cada uma delas, com todos os seus nós (servidor e clientes) e respectivos dados de configuração, constitui um domínio OpenVPN.

A Figura 4.5 representa de forma simplificada o esquema de interligação de N *clusters*, assim como a respectiva decomposição em N-1 ligações VPN ponto-multiponto, cada uma correspondendo a um Domínio OpenVPN (DOVPN). Cada vez que se acrescenta mais um *cluster* à plataforma forma-se um novo DOPVPN que tem como servidor o nó frontal do novo *cluster* e como clientes todos os nós frontais já existentes.

A utilização dos Domínios OpenVPN permite proceder à implementação de cada um deles individualmente, tendo em consideração que para todos domínios os procedimentos são iguais, apenas mudam parâmetros de configuração. Esses procedimentos são:

- Obtenção da chave pública da CA e sua instalação no servidor e nos clientes;
- Geração da chave RSA privada e do certificado digital do servidor OpenVPN e respectiva instalação;
- Geração/obtenção das chaves RSA privadas e dos certificados digitais dos clientes e respectivas instalações;
- Obtenção da chave DH (*Diffie-Hellman*) e sua instalação no servidor e nos clientes;
- Atribuição de rede IP privada de classe C para a interligação de servidor e clientes;

- Identificação da redes privadas do servidor e de cada *cluster* cliente de modo a definir no servidor o respectivo encaminhamento IP;
- Identificação do endereço IP público do servidor;
- Elaboração da configuração do servidor: `n0{N}-srv.conf`, em que N ∈ {2,...,N};
- Elaboração das configurações dos clientes em cada *cluster* pertencente ao domínio: `n0{N}-clt.conf` em que N ∈ {1,...,N-1};
- Activação do processo OpenVPN do respectivo domínio no servidor (`openvpn --config n0{N}-srv.conf`;
- Activação dos processos OpenVPN nos clientes do Domínio OpenVPN, cada um dos quais estabelecerá uma ligação VPN com o processo do servidor (`openvpn --config n0{N}-clt.conf`).

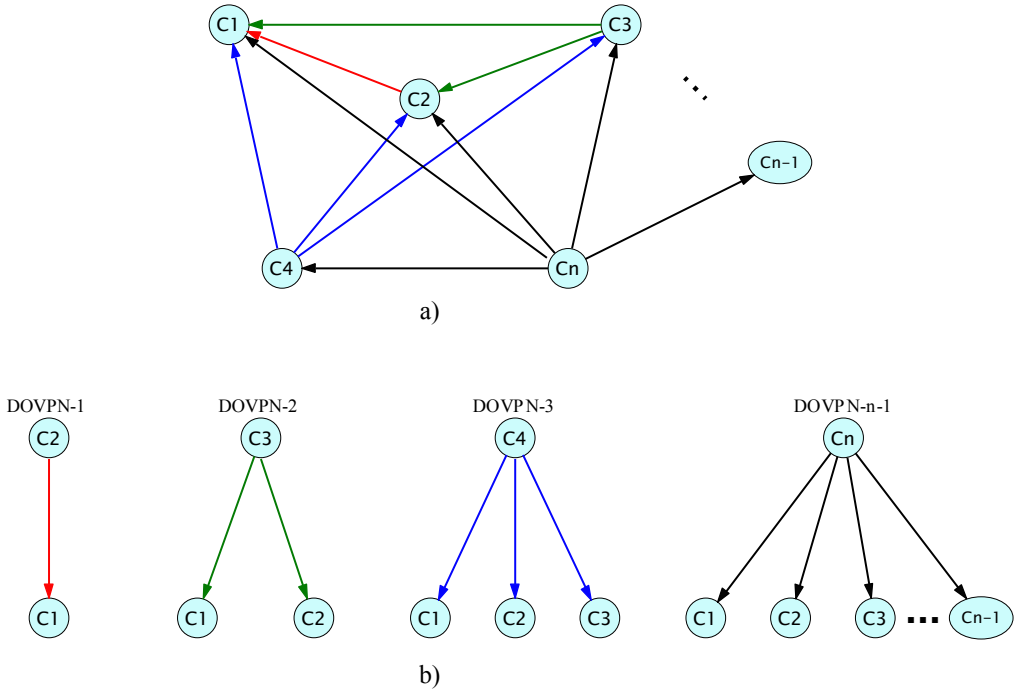

Figura 4.5 – a) Interligação de N *clusters*; b) Decomposição em DOVPN.

Cumpridos todos estes passos tem-se um domínio OpenVPN em funcionamento, ou seja, vários nós frontais ligados como clientes a um servidor VPN, também nó frontal. Procedendo-se de igual forma para todos os Domínios OpenVPN obtém-se a rede de interligação de todos os *clusters* através dos respectivos nós frontais.

É importante realçar que as preocupações de segurança informática subjacentes às arquitecturas de *clusters* baseadas em redes privadas, isoladas do exterior pelos nós frontais, os quais se integram na estratégia de segurança desempenhando a função de *firewalls*, também são consideradas nesta rede de interligação ao garantir-se a confidencialidade e autenticidade da informação transmitida entre *clusters*. Aliás, estas preocupações prevaleceram nesta solução "rede de ligações ponto-multiponto" face à

"rede de ligações ponto-a-ponto", uma vez que desta forma se garante que a informação entre dois *clusters* não circula por outros, mas apenas na ligação VPN (cliente-servidor) que os une.

Resumindo, pretende-se interligar todos os *clusters* que constituirão uma nova plataforma computacional. Esses *clusters* têm uma arquitectura em que os seus nós são interligados por uma rede IP privada, inacessível directamente do exterior. Apenas um nó, frontal, se encontra ligado a uma rede pública. Assim, a solução passa por interligar estes nós frontais usando tecnologia VPN.

Com OpenVPN, software de código aberto baseado em SSL/TLS, implementa-se uma rede de interligação constituída por uma malha de túneis IP ponto-a-ponto entre nós frontais que transpõem eventuais mecanismos de barramento de tráfego, como é o caso dos *firewalls*. O encaminhamento de tráfego IP entre redes privadas passa a ser viável e todos os nós podem comunicar entre si qualquer tipo de informação de forma segura e transparente. Existe no entanto um senão: a tecnologia VPN introduz uma degradação do desempenho da rede de interligação que poderá afectar a execução de aplicações paralelas.

4.3 Sistemas de ficheiros

Nesta secção apresenta-se primeiro, sucintamente, o sistema de ficheiros de *clusters*, particularizando alguns detalhes da implementação Rocks utilizada na plataforma *multi-cluster*. De seguida aborda-se a proposta de arquitectura para um "sistema de ficheiros alargado" que se pretende global em relação à plataforma, descrevendo cada um dos elementos que compõem a solução

No contexto da plataforma *multi-cluster* o acesso a dados e aplicações de uma forma transparente é um outro problema a resolver. Como "juntar" cada um dos sistemas de ficheiros dos *clusters* de forma que utilizadores e aplicações tenham, tanto quanto a possível, uma imagem de sistema uno?

4.3.1 Sistema de ficheiros de clusters

Num ambiente de execução tipo *cluster* existe normalmente uma parte do seu sistema de ficheiros que é comum aos vários nós, de forma partilhada ou distribuída. Existem para este fim várias soluções que são utilizadas neste tipo de ambiente:

- AFS – Andrew File System [38]
- NFS – Network File System [39];
- GPFS – General Parallel File System [40];
- PVFS – Parallel Virtual File System [41].

Estas quatro implementações de sistemas de ficheiros sendo diferentes, têm uma funcionalidade em comum: permitem o acesso à mesma informação (dados ou aplicações) por parte de um utilizador ou aplicação, e a partir de diferentes nós de um *cluster*.

Na configuração mais comum do Rocks é utilizado o NFS, embora também esteja prevista a utilização do PVFS. Neste sistema de ficheiros do Rocks é usado o nó frontal como servidor de ficheiros e os nós de computação são clientes desse servidor. A configuração mais frequente consiste na partilha por NFS de uma partição do sistema de ficheiros do servidor que contém as directorias de trabalho dos utilizadores. É basicamente assim, que num ambiente de *cluster*, utilizadores e aplicações têm a percepção de sistema de ficheiros uno, que por sua vez se reflecte na coerência do ambiente execução. Sempre que um utilizador inicia uma sessão de trabalho num *cluster* a sua directoria fica disponível em todos os nós do *cluster*, podendo ter acesso assim aos seus dados e aplicações.

Para que esta configuração funcione de forma tão transparente é necessário garantir que a directoria de trabalho de cada utilizador tenha a mesma identidade, isto é, o mesmo UID (*User ID*) e o mesmo GID (*Group ID*).

A coerência que assim se exige é, nativamente, promovida pelo NIS (*Network Information System*), um serviço de informação que difunde para os sistemas que compõem um domínio de administração um conjunto de ficheiros de configuração, entre eles o `/etc/passwd` e o `/etc/group`. No caso concreto do Rocks, o NIS é substituído pelo serviço 411, descrito no Capítulo 3 – Tecnologias, que implementa a mesma funcionalidade, mas introduzindo mecanismos de segurança na comunicação.

4.3.2 Sistema de ficheiros alargado

Mas a simplicidade do sistema de ficheiros de um *cluster* perde-se quando se juntam *clusters*. Podem ter-se as interligações física e lógica garantidas, mas isso apenas permite a comunicação directa e bidireccional entre todos os nós da nova estrutura. Estando os *clusters* integrados em domínios de administração diferentes é inviável a fusão dos vários sistemas de ficheiros num único. Tanto mais, que a plataforma *multi-cluster* não deve interferir com o funcionamento autónomo de cada *cluster*.

Assim sendo, a abordagem feita no âmbito deste trabalho procurou encontrar uma solução para um Sistema de Ficheiros Alargado (SFA) a todos os *clusters*, mas que tirasse partido do sistema NFS já instalado em cada um deles.

Propõe-se então um NFS alargado que funciona em paralelo, no sentido de "ao lado de", com o NFS nativo de cada *cluster*. O objectivo é implementar um sistema de ficheiros que permita a utilizadores comuns a vários *clusters* acederem a dados e aplicações transparentemente, como de um acesso local se tratasse. A arquitectura do sistema de ficheiros alargado integra os seguintes componentes:

- Utilizador SFA;
- Directorias de trabalho paralelas;
- Servidores NFS nativos;
- Procuradores NFS;
- Sistema de *Cache* NFS;

O SFA assenta o seu funcionamento essencialmente em três elementos: 1) uma área de disco comum, disponibilizada por todos os nós frontais, que é dividida em fatias

cedidas virtualmente aos utilizadores; 2) procuradores NFS que estabelecem uma correspondência entre cada fatia da área comum e o respectivo utilizador; e 3) um sistema de *cache* hierárquico que minimiza o impacto negativo da rede de interligação de *clusters* da plataforma.

Na Figura 4.6 ilustra-se genericamente a arquitectura do Sistema de Ficheiros Alargado ao *multi-cluster*, com base na qual se passa a descrever cada um dos componentes.

Utilizador SFA

SFA é um utilizador especial, lógico, comum a todos os *clusters* que compõem a plataforma *multi-cluster*, com as mesmas características dos outros utilizadores definidos nos *clusters*.

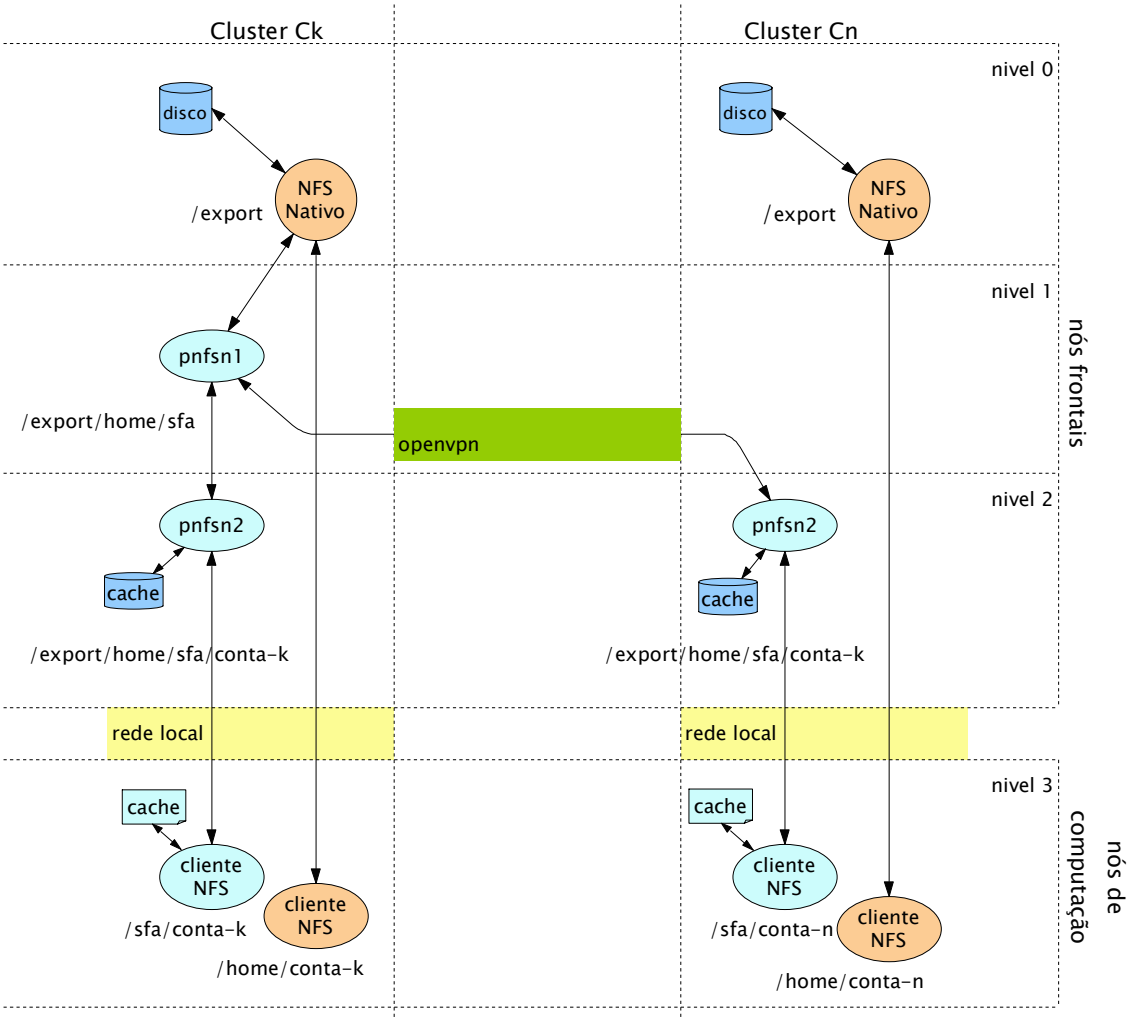

Figura 4.6 – Arquitectura do SFA de uma plataforma de N *clusters*.

A sua directoria de trabalho `/export/home/sfa` constitui a área comum a disponibilizar em todos os nós frontais e que será repartida em *fatias* para os

utilizadores, da forma: `/export/home/sfa/utilizador`. Como é óbvio todas as subdirectorias têm o mesmo `uid` e `gid` do utilizador SFA, uma vez que lhe *pertencem*.

Directorias de trabalho *paralelas*

Os *clusters* Rocks dispõem no nó frontal da partição `/export` para suportar as directorias de trabalho dos seus utilizadores. Esta partição é exportada para nós de computação, sendo montada em cada um deles em `/home`. Concretamente, a directoria de trabalho de um utilizador, fisicamente encontra-se no nó frontal do *cluster* em `/export/home/utilizador` e através do NFS em `/home/utilizador` nos restantes nós do *cluster*.

No âmbito do SFA propõe-se a seguinte estrutura de directorias, *paralela* à estrutura de directorias de trabalho do *cluster*:

- Em nós frontais dos *clusters*: `/export/home/sfa/utilizador;`
- Em nós de computação dos *clusters*: `/sfa/utilizador.`

Em resumo, um utilizador com conta de acesso em vários *clusters* terá duas directorias de trabalho em cada nó de computação:

- `/home/utilizador`: fisicamente reside em NF de cada *cluster*
- `/sfa/utilizador`: fisicamente estará num NF pertencente ao *multi-cluster*.

Ambas as directorias pertencem ao utilizador final e por isso têm os mesmos identificadores: `uid` e `gid`.

Servidores NFS nativos

Estes servidores NFS são os que implementam o sistema de ficheiros de cada *cluster* Rocks e encontram-se instalados nos nós frontais. Normalmente exportam a partição `/export` para todos os nós da rede privada do seu *cluster*.

Exemplo do ficheiro de configuração `/etc/exports` em que 10.1.1.0 representa a rede IP privada do *cluster*:

```
/export      10.1.1.0/255.255.255.0
```

Neste exemplo é permitido o acesso à directoria partilhada aos nós ligados à rede IP privada do *cluster*.

A implementação do SFA implica apenas duas pequenas alterações na configuração do NFS nativo. A primeira diz respeito à política de controlo de acessos à directoria partilhada que passará a incluir `localhost,` com o objectivo de aceitar ligações do próprio nó frontal onde está instalado, como se verá a seguir. No exemplo anterior `/etc/exports` será alterado para:

```
/export      localhost 10.1.1.0/255.255.255.0
```

A outra alteração tem a ver com os portos UDP/TCP associados ao processo `mountd`, os quais são atribuídos dinamicamente. O SFA requer que passem a ser fixos.

À excepção destas duas modificações o SFA é completamente transparente para os servidores de NFS nativos, todos os pedidos, internos do *cluster* e do SFA, são tratados de igual forma.

Procuradores NFS

Uma das questões que se coloca quando um determinado utilizador pretende ter acesso aos seus dados e aplicações distribuídos por vários *clusters*, a partir de qualquer um deles, é compatibilizar as diferentes identidades que esse utilizador assume em cada *cluster*. De facto, como cada *cluster* se insere num domínio de administração diferente é bastante provável que a identidade de um utilizador (UID) e o seu respectivo grupo (GID) varie de *cluster* para *cluster*. Nessa situação, o acesso á informação por NFS a partir de uma única conta de acesso não é viável.

Integrando os Procuradores PVFS, discutidos no Capítulo 3, na arquitectura de SFA, consegue-se mapear as diferentes identidades (UID/GID) de um utilizador nos diferentes *clusters* numa uma única identidade, e assim, garantir o acesso à informação independentemente da conta/*cluster* de onde se faça.

Como se vê na Figura 4.6, nesta arquitectura existem dois tipos de procuradores de NFS: Procuradores NFS de nível 1 (*pnfsn1*) e Procuradores de NFS de nível 2 (*pnfsn2*). Ambos funcionam nos nós frontais dos *clusters*. Não se prevê o funcionamento de procuradores nos nós de computação.

Procuradores de nível 1

Uma vez que se definiu como requisito a coexistência do SFA e dos NFS dos *clusters*, implicaria a abertura dos servidores NFS nativos a todas as redes privadas dos *clusters* da plataforma. Consequentemente, levantar-se-iam dois tipos de problemas: ao nível da administração, exigindo a inclusão das redes privadas nas políticas de controlo de acessos aos servidores NFS; ao nível da segurança, já que o acesso generalizado aos servidores potencia algumas fragilidades intrínsecas ao NFS.

Estes argumentos justificam a existência destes procuradores de nível 1, cuja principal função é autenticar os pedidos RPC com destino aos servidores NFS nativos. Desta forma, estes últimos continuam acessíveis directamente somente às respectivas redes privadas e por outro lado, separa-se a política de controlo de acessos: internos geridos ao nível do NFS e externos geridos pelos Procuradores de Nível 1.

Estes procuradores recebem RCPs dos Procuradores de Nível 2, que correm nos nós frontais dos *clusters*, e encaminham-os para os servidores NFS nativos. Os pedidos são recebidos em portos nível-utilizador, autenticados pelo endereço IP e depois enviados para os portos UDP/TCP do NFS definidos no sistema operativo, 2049 para o `nfsd` e um porto definido no âmbito do SFA para o `mountd` (por exemplo: 33333). Os endereços IP dos nós frontais que são autenticados pertencem aos seus interfaces virtuais, das ligações VPN, uma vez que todo o tráfego que circula entre *clusters* é enviado por estes interfaces.

A interacção com os servidores de NFS é feita sempre em nome do utilizador "Sistema de Ficheiros Alargado", com a conta `sfa` e cuja directoria de trabalho está sedeada em `/export/home/sfa`, a qual é exportada pelos procuradores de nível 1 para os de nível 2.

Neste nível o procurador não faz mapeamento de identidade, isto é, processa pedidos enviados pelos procuradores de nível 2 que já lhe chegam com o UID e GID do utilizador `sfa` e assim os encaminha para os servidores NFS. Por esta razão a configuração do mapeamento é composta por dois pares de UID/GID iguais. Por exemplo:

```
uid  600  600
gid  550  550
```

Os procuradores de nível 1 são permanentes e são activados logo no arranque dos nós frontais dos *clusters*.

Procuradores de nível 2

Estes procuradores actuam em nome do utilizador final e interagem com os procuradores de nível 1, utilizando portos UDP/TCP não privilegiados do sistema. Ao fazerem esta intermediação, operam o mapeamento do utilizador final para o utilizador `sfa`. Exemplificando:

```
uid  620  600
gid  560  550
```

Em que o par UID/GID da segunda coluna representa o utilizador final de um determinado *cluster*, e o par da terceira coluna identifica `sfa` no mesmo ou noutro *cluster*.

Ao contrário dos *pnfsn1* que são permanentes, os *pnfsn2* são activados dinamicamente quando o utilizador inicia uma sessão de trabalho e desactivados quando esta termina. São orientados ao utilizador mas cada instância só pode ligar-se, via um *pnfsn1*, a um único servidor NFS.

Um procurador de nível 2 de um determinado utilizador exporta somente a directoria `/export/home/sfa/utilizador`, que representa a "fatia" desse utilizador num sistema de ficheiros virtual associado à directoria `/export/home/sfa` sedeada num nó frontal. A política de acesso deste *pnfsn2* só permite o acesso à referida directoria do utilizador aos nós do *cluster* onde o procurador está activo. A configuração de *pnfsn2* inclui a seguinte linha:

```
/export/home/sfa/ultilizador  rede_privada_cluster
```

A terminar esta cadeia de ligações, encontra-se o cliente NFS: `mount`, o qual permite estabelecer ligação a *pnfsn2*, exactamente da mesma forma que o faz a um servidor de NFS nativo. Quando este comando é executado nos nós de computação de um *cluster*, monta em cada nó a directoria `/export/home/sfa/utilizador` partilhada por qualquer um dos nós frontais. Esta ligação pode ser representada da seguinte forma:

```
Servidor:/export/home/sfa/conta1 → Cliente:/sfa/conta2
```

Em que `conta1` e `conta2` representam o mesmo utilizador mas com identidades (UIDs) diferentes nos dois *clusters* envolvidos.

No *cluster* em que é montada a directoria remota, o utilizador passa a ter em cada nó a directoria do *cluster* local e a directoria do *cluster* remoto.

Repetindo este processo em todos os *clusters* em que o utilizador tem conta, obtém-se um sistema de ficheiros alargado para o utilizador sem se procederem a alterações nos sistemas de ficheiros dos *clusters*.

O Sistema de Ficheiros Alargado para um dado utilizador pode ter vários servidores ficheiros. No limite, numa plataforma com N *clusters*, poderá ter N servidores, como se exemplifica na Figura 4.7 (onde NF=nó frontal e NC=nó de computação).

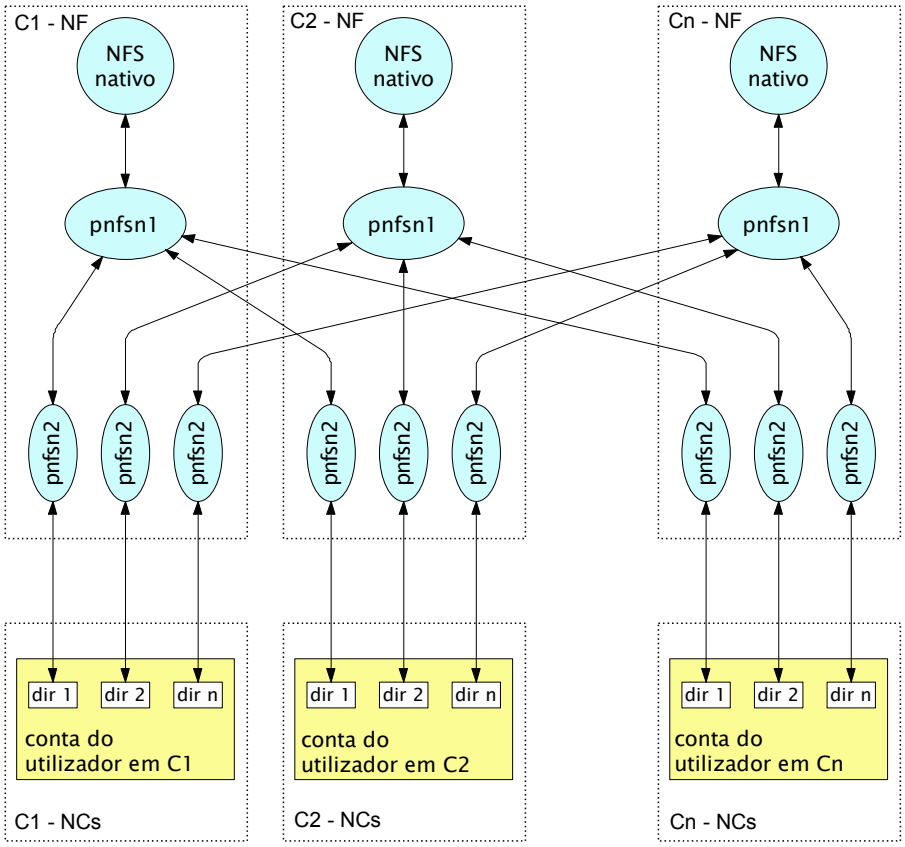

Figura 4.7 – SFA de um utilizador com vários servidores NFS.

Sistema de *cache* NFS

Como já se referiu anteriormente, a utilização da tecnologia VPN tem um impacto negativo no desempenho das comunicações, por aumentar a latência e diminuir a largura de banda, e consequentemente afectando o funcionamento do sistema ficheiros alargado. Este resultado foi comprovado através da realização de algumas medições de leitura e escrita de ficheiros.

Para minimizar este problema propõe-se a utilização de *cache* de disco. Em concreto, utiliza-se uma implementação que integra o pacote de software PVFS. Como na arquitectura SFA se utilizam *pnfs1* e *pnfs2*, em que o primeiro reside no *cluster* que é servidor e o segundo funciona no *cluster* que é cliente, opta-se por activar a

funcionalidade *cache* nos procuradores de nível 2 (*pnfs2*), conforme se mostra Figura 4.6.

Desta forma o SFA implementa um sistema de *cache* de disco dedicada a cada utilizador e por servidor NFS, que acelera o acesso a dados remotos em nós frontais a partir de qualquer nó de computação. Como um *pnfs2* corre num nó frontal de um *cluster*, quando recebe dados do nó frontal remoto que lhe está a fornecer o serviço NFS guarda esses dados localmente antes de os enviar para os nós de computação. Um novo acesso aos mesmos dados, feito pelos mesmos nós de computação, já não requer o acesso ao servidor NFS remoto. Resumidamente, um procurador de nível 2 transfere dados remotos para o nó frontal onde corre, disponibilizando-os a todos os nós de computação onde o utilizador tem sessão activa. Desta forma, num *cluster*, o acesso de nós a dados externos ao *cluster* passa praticamente a ser igual ao acesso de dados locais. O primeiro acesso é mais lento mas os seguintes confirmam a afirmação anterior. Medições efectuadas comprovam estes factos e demonstram que o efeito OpenVPN é praticamente anulado [42].

A *cache* implementada por estes procuradores não guarda as alterações de atributos ou de dados, isto é, escreve directamente no servidor as alterações produzidas nos dados pelos clientes e não as armazena localmente. Pretende-se assim, garantir a consistência dos dados no servidor de ficheiros, garantir que não há dados desactualizados nas diversas *cache*s. Para reforçar a consistência é ainda recomendável configurar cada *cache* para sincronizar periodicamente com o servidor ou então ser forçada a fazer a sincronização por sinalização.

No nó frontal que é servidor de ficheiros de um determinado utilizador, e embora o seu respectivo *pnfsn2* corra nesse mesmo nó frontal, faz sentido activar o sistema de *cache* uma vez que minimiza o tempo de passagem dos dados pelos dois níveis de procuradores. Repare-se que a outra situação corresponde ter os servidores NFS e *pnfsn1* a correr no mesmo nó frontal enquanto *pnfsn2* corre num outro nó frontal. Aqui a *cache* minimiza a utilização da rede de interligação para além do processamento entre servidor e procuradores NFS.

A finalizar, refere-se ainda que o sistema de *cache* em disco será complementado com a utilização de *cache* em memória em cada cliente NFS. Este tipo *cache* funciona num nível hierárquico superior à *cache* em disco e será implementado pelo cliente `mount`.

4.4 Gestão de informação

Independentemente da dimensão da plataforma *multi-cluster* existe um conjunto de informações que é necessário possuir para se proceder às configurações. Uma parte desta informação está disponível nas bases de dados MySQL do Rocks, a outra parte será produzida no âmbito da instalação de novos componentes: OpenVPN, procuradores NFS, entre outros.

À semelhança do Rocks, propõe-se a utilização de uma base de dados MySQL, designada Multicluster, que organiza toda a informação inerente à plataforma *multi-cluster*. Deverá conter informação sobre:

- *Clusters* da plataforma – respectivas redes privadas e endereços IP públicos dos nós fontais;
- Domínios OpenVPN – respectivos servidores e redes IP para interligação;
- Procuradores NFS – gama de portos UDP/TCP e UID/GID de utilizadores SFA.

Alguma desta informação é utilizada de uma forma dinâmica em configurações e na execução de operações.

De entre os nós frontais, um deles poderá ser designado para centralizar a gestão de informação, suportando a base de dados Multicluster e também a PKI (*Public Key Infraestruture*) necessária para o OpenVPN.

4.5 Conclusão

Em conclusão, neste capítulo abordou-se a construção de uma infra-estrutura computacional constituída por vários *clusters* Rocks em diferentes domínios de administração, instalados em entidades geograficamente dispersas, os quais mantém a sua autonomia de funcionamento. A integração destes recursos levanta alguns problemas: 1) como interligar *clusters*, vistos como *ilhas computacionais*, isolados das redes públicas? 2) como integrar sistemas de ficheiros de modo a proporcionar aos utilizadores e aplicações um ambiente igual ao que dispõem no seu ambiente de *cluster*?

Encontraram-se respostas e expuseram-se soluções. Utilização de tecnologia VPN baseada em SSL/TLS para a interligação de *clusters*, e o uso de procuradores de NFS com *cache* de disco integrada para *harmonizar* sistemas de ficheiros distintos e autónomos. Como resultado, cada utilizador com acesso em vários *clusters* passará a dispor de uma nova plataforma constituída por esses *clusters* e dotada de funcionalidades que poderão emular ambientes de programação e execução muito semelhantes aos de um *cluster*.

No entanto outras questões se levantam. Será esta plataforma *multi-cluster* fácil de operar? Estarão os utilizadores interessados numa plataforma tão alargada? Qual a forma para obter o melhor compromisso entre os interesses dos utilizadores e a rentabilidade dos recursos constituintes da plataforma *multi-cluster*?

Capítulo 5

Serviço de *Cluster* Virtual

Tendo como base a plataforma *multi-cluster* tal como apresentada, avança-se agora para uma abordagem que pretende explorar as questões levantadas no final do capítulo anterior.

No essencial apresenta-se uma abordagem que segue a essência das grades computacionais, isto é, que separa utilizadores e aplicações de recursos físicos e domínios de administração. Normalmente, os utilizadores encontram-se associados a sistemas informáticos através de contas de acesso, enquanto dados e aplicações estão *presos* a domínios de administração pelos respectivos sistemas de ficheiros. De certo modo estes obstáculos também estão presentes na plataforma *multi-cluster*. Certamente que um utilizador não terá conta de acesso em todos os *clusters*, por isso continua confinado aos recursos onde tem acesso e, obviamente, pretende usar as mesmas aplicações que utiliza em ambiente de *cluster*.

Seguindo esta orientação, neste capítulo é introduzido o conceito de *cluster virtual* e é apresentada uma estratégia para implementar um serviço que forneça *clusters* virtuais com base nos recursos da plataforma *multi-cluster* e mediante as especificações definidas pelos utilizadores.

5.1 Conceito de *Cluster* Virtual

O *multi-cluster* que atrás se definiu é uma integração de *clusters* heterogéneos distribuídos por diversas instituições e ou domínios administrativos. Até aqui discutiu-se a forma como constituir este ambiente computacional, como interligar *clusters*, como partilhar utilizadores, como partilhar dados e aplicações através da integração de várias tecnologias. No entanto, o objectivo é conceber uma infra-estrutura computacional com serviços acrescentados, de modo a ser usada por diferentes tipos de utilizadores e aplicações e que, ao mesmo tempo, possa tirar partido da heterogeneidade arquitectural dos sistemas envolvidos.

No âmbito deste trabalho pretende-se especificar e desenvolver um outro tipo de *cluster*, definido pelo utilizador de acordo com as suas necessidades, com os seus critérios de agregação de nós, isto é um *Cluster Virtual, definível* por utilizador ou grupo de utilizadores.

O *multi-cluster* propriamente dito, é equivalente à definição de um *cluster alargado*, trans-domínio administrativo que contempla todos os recursos disponibilizados por cada um dos *clusters* componentes, e oferece, por definição, um poder de cálculo agregado máximo.

Contudo, ao nível do utilizador individual, essa capacidade pode ultrapassar as necessidades reais, ou não existir disponibilidade, no momento, visto tratar-se de um ambiente multi-utilizador e multi-aplicação. Por vezes o utilizador poderá ter necessidade de agregar, apenas, um conjunto de nós com características especiais como seria o caso de uma especificação envolvendo 30 nós com bi-processador Pentium 4 e 2 GB de memória, distribuídos pelos diferentes *clusters*.

À configuração obtida como resposta a uma especificação que restringe a configuração máxima de um *multi-cluster* – a um subconjunto de nós e recursos associados – é aqui designada como *Cluster Virtual* (CV). Neste perspectiva, o *multi-cluster* pode ser definido como uma infra-estrutura computacional, sobre a qual, a um nível superior, são redefinidos pelos utilizadores novos *clusters,* obtidos a partir de uma especificação de requisitos que são, durante uma, ou mais sessões, virtualmente *entregues* ao utilizador.

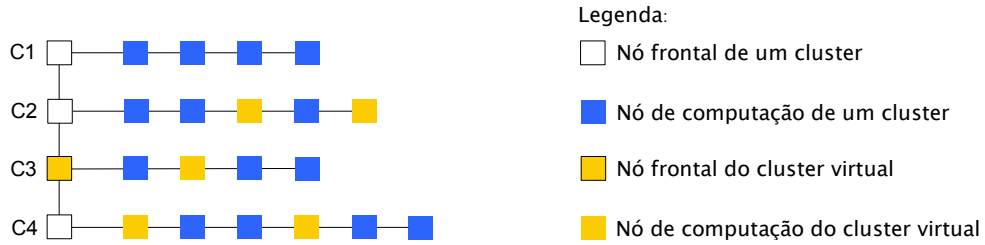

Figura 5.1 – Nós constituintes de um *Cluster* Virtual *entregue* a um utilizador.

O exemplo da figura 5.1 representa um *multi-cluster* (MC) composto por quatro *clusters*, assim pode-se expressar o MC como um somatório de *clusters*:

$$MC = \sum_{i=1}^{4} C_i$$

Por sua vez cada *cluster* C_i pode ser considerado um somatório de nós de computação e do nó frontal (N_0):

$$C_i = \sum_{j=0}^{n} N_{j,i} \text{ sendo } i \text{ índice de } cluster, \in \{1 \text{ a } 4\} \text{ e } j \text{ índice de nó, } \in \{0 \text{ a } n\}.$$

Concretamente, cada *cluster* do exemplo é formado por:

$$C_1 = \{N_{0,1}, N_{1,1}, N_{2,1}, N_{3,1}, N_{4,1}\}$$
$$C_2 = \{N_{0,2}, N_{1,2}, N_{2,2}, N_{3,2}, N_{4,2}, N_{5,2}\}$$
$$C_3 = \{N_{0,3}, N_{1,3}, N_{2,3}, N_{3,3}, N_{4,3}\}$$
$$C_4 = \{N_{0,4}, N_{1,4}, N_{2,4}, N_{3,4}, N_{4,4}, N_{5,4}, N_{6,4}\}$$

Concluindo a exemplificação do conceito de *cluster* virtual, verifica-se que da especificação feita pelo utilizador, resultou o seguinte *cluster* virtual (nós a cor clara):

$$CV = \{N_{0,3}, N_{3,2}, N_{5,2}, N_{2,3}, N_{1,4}, N_{4,4}\}$$

Ou seja, um *cluster* composto pelo nó frontal do custer 3, pelos nós 3 e 5 do *cluster* 2, pelo nó 2 do *cluster* 3 e pelos nós 1 e 4 do *cluster* 4.

Resumindo, de vários *clusters* distribuídos constrói-se um *multi-cluster* base que depois, de forma dinâmica, dá origem a *clusters* virtuais – definidos pelo utilizador para correr as suas aplicações.

5.2 Aproximação ao ambiente *Grid*

O conceito *Cluster* Virtual (CV) introduzido acima conduz-nos ao ambiente *Grid*, com partilha de recursos flexível, segura e coordenada entre conjuntos dinâmicos de indivíduos, instituições e recursos [43].

A aproximação à *Grid* que neste trabalho se pretende introduzir, não se faz adoptando modelos já estabilizados ou normalizados, como o caso do modelo OGSA (*Open Grid Services Arquitecture*), mas mais pela assimilação de conceitos e práticas que caracterizam as implementações de grades computacionais. Assim esta abordagem à *Grid* que aqui é feita assenta em dois vectores: 1) a forma de utilização e 2) o vínculo do utilizador com a infra-estrutura computacional.

A utilização de *Grids* é caracterizada pela facilidade de acesso, com uma interacção simplificada, muito apoiada em portais Web. Também neste projecto se utiliza essa facilidade, sobretudo no que respeita à configuração do CV e também no estabelecimento de sessão interactiva, usando o mínimo de recursos, a partir de qualquer computador ligado à Internet. Daí a existência de um portal que corporiza esta intenção e que se aborda adiante.

O outro vector também explorado nesta solução, é então o vínculo do utilizador com a plataforma computacional, que passa pela forma como são atribuídos e associados os recursos ao utilizador. Nas *Grids* a atribuição de recursos é dinâmica e a sua associação ao utilizador passa por processos de autenticação e autorização baseados em certificados digitais.

Na presente solução o utilizador não necessita de ter qualquer conta de acesso em nenhum *cluster*. Apenas é exigido que faça o seu registo, e após a validação superior deste, poderá usar o Serviço de *Cluster* Virtual, sendo a atribuição e autorização de recursos realizadas usando mecanismos de autenticação, com chave pública ou certificados digitais. Mas aqui reside uma particularidade da solução, a associação entre os recursos físicos do CV e o seu respectivo utilizador é feita através de *utilizadores lógicos* (UL) existentes em todos os *clusters* e cuja caracterização é feita mais à frente.

A flexibilidade subjacente à *Grid* Computacional levanta problemas cuja resolução exige uma coordenação de esforços das várias administrações de domínios envolvidas para a sua concretização.

De facto a implementação de um serviço deste tipo levanta questões como: Quem são os utilizadores deste serviço e como gerir o seu acesso? Como atribuir recursos no contexto do serviço? Como restringir o acesso para além dos recursos atribuídos e garantir o normal funcionamento de cada *cluster* integrante do *multi-cluster*? Como

passar dos recursos físicos atribuídos para a virtulaziação do *cluster* de um determinado utilizador? Que interacção estabelece o utilizador com o *seu cluster* virtual? Como (re)criar o seu ambiente de trabalho, para dessa forma proceder à execução das suas aplicações? Como virtualizar o ambiente de programação?

Obviamente que as respostas a estas perguntas só se concretizam enriquecendo a plataforma *multi-cluster* com funcionalidades que a tornem flexível e dinâmica, como se espera de uma *Grid*. Tal implica, uma vez mais, integrar tecnologias mas também desenvolver ferramentas que permitam acima de tudo gerir os recursos de cada *cluster* real de uma forma transparente, sem prejudicar o seu funcionamento autónomo, de tal modo que o *Utilizador de Cluster Virtual* (UCV) seja considerado pelo sistema de gestão de cada *cluster* como mais um utilizador. Sucintamente, a solução passa por introduzir uma camada de abstracção, intermediária entre o UCV e o *multi-cluster*.

Nas secções seguintes é feita a especificação desta camada intermediária, assim como é descrito o seu funcionamento.

5.3 Portal de acesso ao serviço

É sem dúvida a face visível da solução global e desempenha a função que é, de certo modo, muitas vezes associada aos ambientes *Grid*. É através deste portal que se acede ao Serviço *Cluster* Virtual, logo é aqui que o utilizador realiza praticamente todas as tarefas que depois lhe irão permitir usar os *seus* recursos computacionais.

As funcionalidades deste portal são simples e podem resumir-se a três:

- Registo de utilizadores;
- Negociação de recursos;
- Estabelecimento de sessão de trabalho.

Obviamente, a cada a uma destas funções correspondem sub-funções, mas o importante é realçar que a interactividade do utilizador com o serviço passa pelas fases implícitas naquelas três funções que se podem detalhar de imediato.

5.3.1 Registo de utilizadores

Caracterizando genericamente os utilizadores deste serviço, poder-se-á considerar que são de alguma forma especialistas de diferentes áreas do conhecimento que em certos momentos necessitam de recursos computacionais para desenvolverem o seu trabalho. Para além desta definição genérica, podem-se diferenciar quanto aos recursos que à partida já dispõem. Assim sendo, é provável que existam utilizadores: 1) que pelas mais variadas razões, não possuem quaisquer recursos e que pontualmente necessitem de recorrer a recursos computacionais, ou 2) que já usam *clusters* mas a partir de uma determinada altura necessitam de complementar esses recursos com outros integrados na plataforma *multi-cluster*, por possuírem outras características ou pelo simples facto de aumentarem capacidade de processamento. Ambas as situações se enquadram no serviço aqui proposto.

Para que qualquer uma destas utilizações aconteça é necessário que os utilizadores procedam ao seu registo no serviço, fornecendo num formulário Web a seguinte informação:

- Pessoal – nome do utilizador, estatuto, área de trabalho, instituição e enquadramento de utilização;
- Recursos próprios – identificação de quais e as respectivas contas de acesso, caso o utilizador as tenha. Para utilizar o Serviço de Cluster Virtual não é necessário possuir conta de acesso em qualquer cluster;
- Credenciais – indicação da respectiva senha para futuro acesso ao portal. O utilizador deve ainda enviar a sua chave pública, que normalmente usa para se autenticar em outros serviços, e que será usada também para aceder aos CV;

Neste processo de registo é gerado automaticamente um par de chaves RSA, privada e pública, que ficará associado ao utilizador, identificando-o em determinadas operações realizadas no contexto do SCV.

Esta informação introduzida via portal e as chaves RSA atribuídas são armazenadas na base de dados "Multicluster" em MySQL, a qual já foi referida no capítulo anterior, numa tabela dedicada ao registo de utilizadores.

Ao utilizador registado é-lhe atribuído um identificador UCV_i, Utilizador de *Cluster* Virtual, em que *i* é um inteiro de 1 a N. Assim, perante o SCV o utilizador terá uma identidade que pode ser representada da seguinte forma:

UCV_i = {Chave pública própria, Chaves RSA (pública e privada) internas}

Após este registo o utilizador fica habilitado a aceder ao Serviço de *Cluster* Virtual, propriamente dito, indicando quais as suas credenciais, ou seja, o seu identificador e respectiva senha.

5.3.2 Negociação de recursos

Depois de registado, o utilizador acede ao Serviço *Cluster* Virtual (SVC) através do portal autenticando-se aí com as credenciais que definiu no registo. De seguida solicita os recursos disponíveis no *multi-cluster*, desencadeando-se um pedido ao escalonador de todos os *clusters* sobre o estado dos recursos. Como resposta o escalonador devolverá informação sobre os nós ocupados, os nós livres e a ainda poderá fornecer as características de cada nó. Esta informação é tratada, formatada em tabela e apresentada ao utilizador no portal que, de acordo com as suas necessidades, selecciona:

- Nós de computação para o seu CV;
- Nó frontal para o CV, que será um nó frontal dos clusters que integram o multi-cluster.

O utilizador submete esta selecção de recursos ao SCV e, sendo aceite, dará origem à geração do seu *Cluster* Virtual. Em caso contrário, deverá ser efectuada nova iteração para identificar recursos disponíveis. Esta operação é crítica porque deve ter em conta a dinâmica de cada *cluster*. Entre o momento em que o escalonador informa sobre o estado do seu *cluster* e o momento em que o utilizador submete a sua selecção de

recursos poderá decorrer um período temporal suficiente para que o estado de utilização dos *clusters* se tenha alterado, passando a existir mais ou menos recursos.

A aceitação dos recursos seleccionados pelo utilizador implica de imediato a reserva de cada um desses recursos, usando-se o escalonador do respectivo *cluster*.

O resultado de uma possível negociação/atribuição de recursos pode ser o CV_3 com quatro nós, representado abstractamente da seguinte forma:

$$CV_3 = \{N_{0,2}, N_{3,2}, N_{2,6}, N_{10,5}\}$$

Em que $N_{0,2}$ é o nó frontal do *cluster* 2 e é, neste contexto, também o nó frontal de CV_3. $N_{3,2}$ é o nó 3 do *cluster* 2 e assim sucessivamente.

Todo o restante processo de constituição do CV é transparente para o utilizador, terminando quando lhe for apresentado, automaticamente, uma sessão de trabalho no nó frontal do CV, estabelecida através de um cliente SSH.

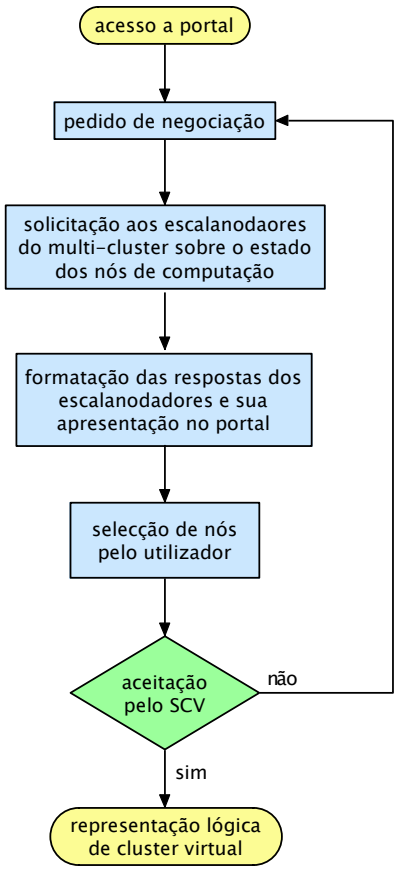

Figura 5.2 – Diagrama de fluxo da negociação de recursos.

5.4 Utilizadores lógicos

A par do portal o Utilizador Lógico (UL) constitui outro elemento intrínseco ao Serviço de *Cluster*. Normalmente, um utilizador real é representado por uma conta de acesso que só ele pode usar. Um UL consiste num utilizador fictício de um *cluster*, não representando ninguém, ao qual corresponde uma conta de acesso do tipo $UL_{i,c}$, em que *i* representa o identificador de UL no *cluster* cujo identificador no *multi-cluster* é *c*. Exemplificando, a conta $UL_{2,10}$ representa o UL número 2 pertencente ao *cluster* 10 da plataforma *multi-cluster*.

Cada uma destas contas, não tendo um utilizador definido, pode então ser cedida temporariamente a um utilizador efectivo para aceder a recursos de um *cluster*. Terminada essa utilização a conta de acesso fica de novo disponível.

Cada *cluster* disponibiliza um conjunto destas contas genéricas para utilização no contexto do SCV, constituindo assim uma infra-estrutura de contas de acesso, a qual é gerida dinamicamente, a partir de uma tabela "utilizadores_logicos" que integra a base de dados "Multicluster", contendo a seguinte informação:

- Cluster físico;
- Nome da conta;
- Identificador de conta UID/GID;
- Portos UDP de entrada/saída do Procurador de nível 2;
- Estado de utilização (livre/ocupado).

Para cada CV é efectuado o *aluger* de um conjunto de contas UL ao utilizador, que lhe permitem o acesso aos nós do seu CV. Usando o CV exemplo da secção anterior, CV_3, pode agora supor-se que os respectivos UL são:

$$CV_3 = \{UL_{4,2}, UL_{9,6}, UL_{3,5}\}$$

É de notar que embora CV_3 seja constituído por 4 nós, apenas são necessários três UL, uma vez que dois desses nós, $N_{0,2}$ e $N_{3,2}$, pertencem ao mesmo *cluster* 2. Apenas é necessário um UL por *cluster*. $UL_{9,6}$ será a conta de acesso para o *cluster* 6 enquanto $UL_{3,5}$ será a conta para aceder aos nós do *cluster* 5.

Resumindo, os recursos atribuídos a CV_3, nós e ULs, são representado pelos seguintes conjuntos:

$$CV_3 = \{N_{0,2}, N_{3,2}, N_{2,6}, N_{10,5}\}$$
$$CV_3 = \{UL_{4,2}, UL_{9,6}, UL_{3,5}\}$$

5.5 Representação lógica de CV

A selecção de recursos efectuada pelo utilizador é de facto o ponto de partida para a configuração do CV e consequentemente para a sua implementação. O que realmente o utilizador definiu foi apenas quais os nós que compõem o seu CV. No entanto é

necessário obter mais informação e contextualizá-la, formatá-la, para que funcione como a representação lógica do CV. Ou usando uma expressão redundante, trata-se da representação virtual de CV. Será a partir desta representação que se fará a configuração de todo o CV.

A representação lógica tem dois tipos de informação: 1) sobre o nó frontal de CV e 2) sobre cada *cluster* físico que participa na formação de CV; que se detalham de imediato.

1) Informação do nó frontal de CV_i

- Nome;
- Endereços IP;
  - Privado;
  - Público;
- *Cluster* físico (Ci);
  - Rede IP privada;
  - Mascara rede IP privada;
- Conta de partilha (`sfa`) para o Sistema de Ficheiros;
  - Identificador de utilizador (uid);
  - Identificador de grupo (gid);

2) Informação de cada *cluster* participante em CV_i

- *Cluster* físico (C_i);
  - Rede IP privada;
  - Mascara rede IP privada;
- Utilizador Lógico
  - Identificador de utilizador (uid);
  - Identificador de grupo (gid);
- Procurador NFS Nível 2 (pnfsn2);
  - Porto UDP de `mountd`;
  - Porto UDP de `nfsd`;
- Nó (N_{ji})
  - Endereço IP;
- Escalonador;

Cada *cluster* participante num CV contribui com: 1) um ou mais nós de computação, 2) um Utilizador Lógico e 3) um Procurador NFS de nível 2. Um destes *clusters* ainda fornece o nó frontal que também é o seu.

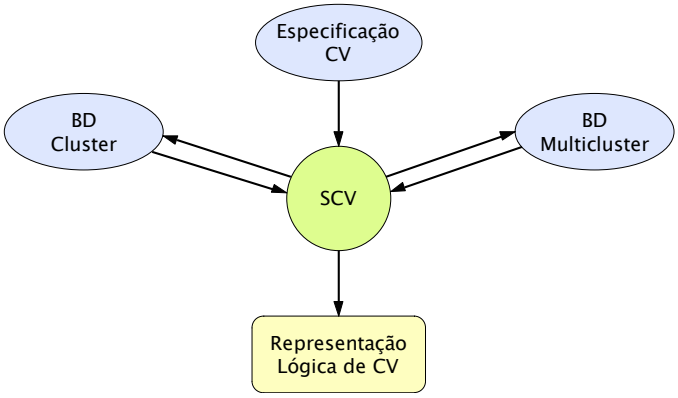

Figura 5.3 – Fontes de informação para a Representação Lógica de CV.

A recolha de informação para a representação lógica é feita com base na especificação feita pelo utilizador na fase da negociação de recursos, e usando-a depois para consultar as bases "Cluster" do Rocks e "Multicluster" da plataforma *multi-cluster*, como se ilustra na Figura 5.3.

Recuperando o exemplo de CV_3, a sua representação lógica poderia ser:

no_frontal
cluster = C2
rede_ip_priv = 10.2.2.0
rede_ip_priv_mask = 255.255.255.0
nome = N0_2
ip_pub = 193.136.35.6
ip_pri = 10.2.2.1
sfa = `/export/home/sfa/ul4_2`
uid = 600
gid = 550

***cluster* C2**
cluster = C2
rede_ip_priv = 10.2.2.0
rede_ip_priv_mask = 255.255.255.0
no = N3_2
ip = 10.2.2.23
utilizador_logico = ul4_2
uid = 600
gid = 550
pnfsn2
porto_nfsd = 51600
porto_mountd = 52600
escalonador = PBS

***cluster* C6**
cluster = C6
rede_ip_priv = 10.6.6.0

rede_ip_priv_mask = 255.255.255.0
no = N2_6
ip = 10.6.6.253
utilizador_logico = ul9_6
uid = 612
gid = 542
pnfsn2
porto_nfsd = 51612
porto_mountd = 52612
escalonador = PBS

***cluster* C10**
cluster = C10
rede_ip_priv = 10.10.10.0
rede_ip_priv_mask = 255.255.255.0
no = N10_5
ip = 10.10.10.23
utilizador_logico = ul3_5
uid = 598
gid = 532
pnfsn2
porto_nfsd = 51598
porto_mountd = 52498
escalonador = PBS

Com base na representação lógica de CV é possível então partir para a virtualização do ambiente de CV.

5.6 Virtualização do ambiente de *Cluster*

Esta fase corresponde no essencial à criação, propriamente dita, do *Cluster* Virtual e pode decompor-se em três partes: 1) construção do sistema de ficheiros de CV, 2) autenticação e segurança de CV e 3) acesso remoto a CV.

5.6.1 Sistema de ficheiros

Como se viu na primeira parte, a plataforma *multi-cluster* dispõe de um Sistema de Ficheiros Alargado (SFA) que irá ser usado neste contexto de *Cluster* Virtual. Em termos de sistema de ficheiros o que realmente se pretende é reproduzir sobre a plataforma *multi-cluster* o ambiente de execução de um *cluster* típico. Ou seja, um nó frontal que, entre outros serviços, é servidor de ficheiros usando NFS. Neste caso do CV, tendo-se definido qual o servidor de ficheiros, é necessário garantir o acesso por NFS a este, de todos os nós do CV, independentemente do *cluster* físico onde se encontrem. Utilizando o SFA é apenas necessário usar os procuradores NFS de nível 2, isto é, activar um procurador por *cluster* participante no CV. Por outras palavras, isto

significa que será activado um procurador de nível 2 por cada UL do CV. Como já se referiu no Capítulo 4, a activação dos procuradores é feita no nó frontal de cada *cluster*.

Configuração de Procuradores NFS

Um CV terá tantos procuradores NFS de nível 2 (*pnfsn2*) quanto o número de *clusters* que participam na sua configuração. Como são orientados ao utilizador, UL_{ic} neste contexto, podem ser designados por $pnfsn2.ul_{ic}$. A configuração de cada um destes *pnfsn2* é feita com base na representação lógica do CV e é diferente de *cluster* para *cluster*, há no entanto informação que é comum, concretamente, a directoria exportada e respectivos `uid/gid` em que assenta o sistema de ficheiros de CV. Todos os *pnfsn2* acedem a esta directoria. Outra informação que é especifica de cada procurador:

- Rede IP de cada cluster e respectiva máscara;
- UL em cada cluster participante de CV e respectivos uid/gid.

Com esta informação constroem-se, dinamicamente, os ficheiros de configuração `exports_mount`, `exports_nfsd` e `map` de cada UL.

Activação de Procuradores NFS

Após definidas as configurações é necessário activar cada um dos procuradores $pnfsn2.ul_{ic}$, o que implica a execução de dois processos no nó frontal de cada *cluster* participante no CV: `pvfs.nfsd` e `pvfs.mountd`. Mais uma vez é preciso obter informação para parametrizar cada um dos processos, nomeadamente:

- Endereço IP privado do nó frontal e respectiva máscara;
- Portos UDP de entrada dos processos pvfs.mountd e pvfs.nfsd de pnfsn1 a correrem no nó frontal de CV;
- Portos UDP de saída dos processos pvfs.mountd e pvfs.nfsd de pnfsn2.ulic a correrem em todos nós frontais dos clusters que participam em CV.

Estas tarefas de parametrização e activação de processos são executadas remotamente nos diversos sistemas e de forma dinâmica.

Cada um destes procuradores $pnfsn2.ul_{ic}$ vai estabelecer uma ligação com o procurador NFS de nível 1 em execução no nó frontal do CV, que já se encontra em funcionamento. Relembra-se que estes procuradores de nível 1 são activados logo no arranque do sistema operativo de cada nó frontal.

De realçar que cada $pnfsn2.ul_{ic}$ utiliza o endereço IP privado do nó frontal de CV para que a ligação que estabelece com *pnfsn1* se faça através dos túneis VPN da rede de interligação. Desta forma contornam-se eventuais problemas que resultariam pelo facto de nestas ligações entre procuradores se usarem portos UDP muito elevados, os quais normalmente se encontram bloqueados nos *firewalls* das instituições onde se encontram sedeados os *clusters*. Aliás, no contexto do *multi-cluster* os nós frontais são sempre acedidos através dos seus endereços IP privados com o objectivo de encaminhar todas as comunicações entre *clusters* através dos túneis IP.

Como se referiu no Capítulo 4, no SFA utiliza-se um sistema de *cache* hierárquico, com dois níveis, em que o primeiro nível consiste numa *cache* de disco intrínseca aos

procuradores NFS. Assim, os *pnfsn2.*ul_{ic} são activados com a funcionalidade *cache*. O segundo nível de *cache*, de memória, é activado nos clientes NFS.

Activação de clientes NFS

Para concluir o sistema de ficheiros do CV só falta activar os clientes NFS, os quais montam nos sistemas de ficheiros locais dos nós do CV a directoria exportada pelo nó frontal. Por outras palavras, os clientes NFS estabelecem uma ligação entre a directoria local de cada nó, `/sfa/ul`$_{ic}$, e a directoria remota, `/export/home/sfa/ul`$_{jc}$, sendo `ul`$_{ic}$ e `ul`$_{jc}$ utilizadores lógicos diferentes para o nó frontal e para os nós de computação, respectivamente.

Estes clientes de NFS são activados dinamicamente pelo SCV que executa o comando `mount` devidamente parametrizado. Terminada com êxito esta operação, estará também concluído o sistema de ficheiros de CV.

5.6.2 Autenticação e segurança

Uma parte do ambiente de *cluster* está virtualizada. Há um nó frontal e um conjunto de nós de computação geograficamente distribuídos que partilham por NFS um sistema de ficheiros, como acontece num normal *cluster*, com a particularidade de que este foi *desenhado* por um determinado utilizador com a finalidade de aí executar as suas aplicações paralelas.

Directoria de trabalho de UCV

O SFA implementa duas directorias de trabalho *paralelas*. Isto significa que cada UL que compõem um CV tem duas directorias de trabalho: a que pertence ao *cluster* físico, `/export/home/ul`$_{ic}$, e a que pertence ao CV, `/sfa/ul`$_{ic}$. Sendo a primeira, a directoria de trabalho que se encontra definida em `/etc/passwd`, e por isso, a directoria definida no ambiente de trabalho. Tem-se portanto, vários UL de um CV, cada um com a sua directoria de trabalho *real* e pretende-se que todos comutem a sua directoria de trabalho para a mesma directoria *virtual* do UCV, exportada pelo nó frontal de CV. Isto implica alterar em `/etc/passwd` de todos os *clusters* participantes no CV, a directoria de trabalho do seu respectivo UL, de modo a ser `/sfa/ul`$_{ic}$. Desta forma, CV é composto por vários UL mas todos com a sua directoria de trabalho sintonizada na directoria exportada pelo nó frontal do CV.

Na prática esta operação é realizada pelo SCV ao nível de cada nó frontal dos *clusters* Rocks e em que intervém o seu serviço de informação 411, difundindo a alteração aos restantes nós. Concluída esta operação, a directoria de trabalho de todos os UL de CV será a mesma: `/sfa/ul`$_{ic}$.

Acesso ao nó frontal de CV

Uma outra questão está relacionada com o acesso ao CV versus autenticação. O CV existe mas ainda não é possível usá-lo por não haver nenhum acesso definido. Não existe nenhuma associação entre o CV e o utilizador que o *desenhou* e agora o pretende utilizar. A associação permitirá autenticar o utilizador, dando-lhe acesso ao seu CV.

Aquando do seu registo, o utilizador enviou para o SCV a sua chave pública própria. De modo a garantir o acesso externo CV por SSH, como é objectivo, a chave pública do utilizador é colocada na directoria de trabalho do UL atribuído para o nó frontal do CV. Desta forma é garantido acesso por autenticação com base na chave privada do utilizador que estará na sua posse. Assim, usando uma conta UL que não precisará de conhecer, o utilizador acede ao nó frontal de CV usando autenticação baseada em chaves públicas.

Acesso restrito aos nós de computação

Depois de se garantir o acesso ao nó frontal, o acesso aos restantes nós de computação passaria por replicar a chave pública do utilizador para esses nós e colocar a sua chave privada no nó frontal. No entanto, tendo em consideração que a conta UL no nó frontal só é atribuída ao utilizador temporariamente, é importante garantir que terminada a sessão de trabalho a chave privada é eliminada. Havendo uma falha poderá ser comprometida a confidencialidade da chave privada. Assim, para não se correr este risco prefere-se a solução que utiliza as chaves RSA internas, geradas aquando do registo do utilizador. Colocando esse par de chaves RSA na mesma directoria de trabalho de UL no nó frontal, e uma vez que esta directoria é partilhada por NFS para os nós, é possível autenticar o acesso a qualquer nó de CV com base nas chaves RSA internas. Aliás este funcionamento é semelhante ao que se verifica num *cluster* Rocks. Quando um utilizador acede pela primeira vez a um *cluster* Rocks, é gerado um par de chaves RSA, `id_rsa.pub` e `id_rsa`, que ao ficar na sua directoria de trabalho partilhada por NFS para todos os nós, permite o acesso a estes, por exemplo, via SSH.

Resumindo, a autenticação do acesso aos recursos do CV é feita da seguinte forma:

- No acesso do exterior ao nó frontal, utiliza-se autenticação com base na chave pública do utilizador;
- No acesso entre quaisquer nós do CV, utiliza-se autenticação com base em chaves RSA internas do utilizador.

Garantir que o acesso é restrito ao conjunto de nós de CV é uma condição fundamental para a segurança de cada *cluster* físico. Normalmente um utilizador de um *cluster* tem acesso a todos os nós. O que também acontece com os UL de cada *cluster*. Um UL é utilizador normal e como tal terá acesso a todos os nós do seu *cluster*. No entanto quando inserido num CV a sua directoria de trabalho é comutada para o sistema de ficheiros de CV. Assim sendo, como qualquer acesso com autenticação por chave pública exige a presença desta chave na directoria de trabalho, tal condição só se verifica nos nós pertencentes ao CV, nos quais foi montada a directoria de UCV, exportada pelo nó frontal de CV, que contém as chaves internas RSA. Nos restantes nós não existe a directoria `/sfa/ul`$_{ic}$, consequentemente, a autenticação falha. Nestes nós UL acaba por não ter directoria de trabalho, uma vez que a directoria que está definida em `/etc/passwd` não existe porque não foi montada.

A conjugação de 1) autenticação com base em chaves públicas e 2) comutação das directorias de trabalho dos ULs que integram um CV, evita a utilização abusiva de outros recursos da plataforma *multi-cluster* que não tenham sido atribuídos a esse CV.

5.6.3 Acesso remoto a CV

Seguindo a filosofia da *Grid* é vantajoso que o utilizador possa aceder ao seu CV de qualquer parte e bastando, para isso, dispor de um programa de navegação na Internet como o *Firefox* ou *Internet Explorer*. Algo como ter sempre um terminal virtual para estabelecer uma sessão interactiva com o seu CV, por exemplo, em SSH.

A proposta passa por utilizar um cliente SSH que é descarregado automaticamente quando se acede ao Portal do SCV e é executado no computador do utilizador. Antes, o SCV elabora a página Web que contem o programa já devidamente configurado para estabelecer ligação com o nó frontal, especifico, do CV do utilizador. Para além do endereço IP deste nó, a parametrização inclui a conta UL onde previamente foi colocada a chave pública do utilizador. Este UCV apenas tem de indicar ao cliente SSH onde se encontra a sua chave privada.

Com a sessão estabelecida é ainda possível usar módulos integrados no cliente SSH, como o SCP ou o SFTP, para efectuar transferências de dados e aplicações do exterior para o CV.

5.6.4 Execução de aplicações paralelas

Constituído o CV como ambiente de execução o utilizador quererá correr as suas aplicações paralelas desenvolvidas em ambientes de programação como PVM [44] ou MPI [45], os quais requerem o conhecimento prévio do ambiente de execução, definido habitualmente em ficheiros de configuração.

Em ambiente de *cluster* o utilizador tem acesso directo aos recursos, e com a ajuda do escalonador obtém informação para elaborar a configuração do seu ambiente de execução. A informação relevante para este fim é constituída por: nome/endereço dos nós a usar, contas de acessos a esses nós e respectivas senhas, directorias de trabalho e das aplicações.

No CV, parte desta informação é fornecida pelo SCV disponibilizando um ficheiro com a indicação dos endereços IP dos nós de computação e respectivos UL. Poderá também fornecer indicação sobre a localização dos programas que implementam os ambientes de programação. Mas tendo em consideração a eventual heterogeneidade de *clusters*, especialmente no diz respeito a versões de software, será aconselhável transferir para o CV este tipo de programas. Em relação a senhas de contas não é necessário informação uma vez que se utiliza autenticação com chaves públicas.

Com esta informação o utilizador pode configurar o seu ambiente de execução. Por exemplo, para configurar o ambiente PVM, basta produzir o ficheiro `hostfile` com indicação dos endereços IP dos nós e respectivas contas e directorias de trabalho. Definindo por último as variáveis de ambiente de sistemas (por exemplo: `PVM_ROOT` e `PVM_ARCH`), o utilizador dispõe de recursos aptos a correr as suas aplicações paralelas.

5.7 Conclusão

Concluindo este capítulo, relembra-se que a plataforma *multi-cluster* apresentada no Capítulo 4 é uma infra-estrutura computacional constituída por *clusters* Rocks interligados por tecnologia OpenVPN. É ainda caracterizada por implementar um Sistema de Ficheiros Alargado baseado numa hierarquia de procuradores NFS complementada por um sistema de *cache* que melhora o desempenho do sistema de ficheiros distribuído. Apesar destas características a plataforma *multi-cluster* não é um *produto acabado*, por herdar a organização de um *cluster*. Na prática este *multi-cluster* é um *cluster* alargado com utilizadores convencionais e com uma administração repartida entre a administração de domínio local e uma administração transversal encarregue de gerir a rede de interligação e a estrutura de procuradores NFS.

O SCV descrito neste capítulo, acrescenta à plataforma flexibilidade que a aproxima da filosofia de *Grid* Computacional, ao separar utilizadores da infra-estrutura física introduzindo utilizadores lógicos, promovendo a autenticação de utilizadores e recursos através de certificados digitais e ao permitir uma forma de utilização orientada à Web.

Para além destas semelhanças com a *Grid*, o funcionamento de SCV é diferente. Enquanto na grade computacional se submetem trabalhos que depois são executados algures, com o SCV submetem-se especificações de *clusters* e obtém-se CV, nos quais se podem correr directamente aplicações, como se tratassem de *clusters* físicos com o mesmo ambiente de execução. Isto permite ao utilizador correr as suas aplicações, com as suas particularidades, sem ter de as alterar.

Os ambientes de programação ainda se encontram intrinsecamente identificados com os ambientes de execução. As aplicações desenvolvidas para esses ambientes não são facilmente portáveis para ambientes de grades computacionais.

Assim, o SCV é uma solução intermédia entre a solução clássica de *cluster* dedicado a grupos de utilizadores ou aplicações e a grade computacional orientada a grandes comunidades de utilizadores agrupados em Organizações Virtuais.

Capítulo 6

Implementação

O Capítulo 6 é dedicado à exposição de pormenores de implementação, tendo como objectivo demonstrar a viabilidade das propostas efectuadas nos Capítulos 4 e 5. Não se pretende aqui apresentar uma solução integral e final, mas antes reproduzir partes do trabalho laboratorial realizado ao longo do tempo, que consistiu em estudar e testar tecnologias e definir estratégias para as integrar. No Capítulo 3 é feita uma explanação das tecnologias mais relevantes e o contexto da sua aplicação.

6.1 Introdução

Utilizando uma plataforma de desenvolvimento virtual, isto é, baseada em máquinas virtuais VMware, são percorridas as várias fases, primeiro para a constituição da plataforma *multi-cluster*, depois para a obtenção de um *cluster* virtual (CV).

Na constituição do *multi-cluster* é instalado e configurado o OpenVPN [46], destacando aspectos referentes à autenticação com certificados digitais, assim como aspectos relacionados com encaminhamento de tráfego IP. No âmbito do Sistema de Ficheiros Alargado, é abordada a instalação dos procuradores NFS de níveis 1 e 2.

Em relação ao funcionamento do Serviço de *Cluster* Virtual, é realizado o *percurso* para a obtenção de um CV, destacando-se a implementação do sistema de ficheiros de CV, a associação da identidade do utilizador real ao CV e o estabelecimento de sessão SSH.

Os *clusters* utilizados são implementados com o software NPACI Rocks e com uma configuração simples. Ainda antes de se avançar, relembra-se a arquitectura deste tipo de *clusters Beowulf* através da Figura 6.1.

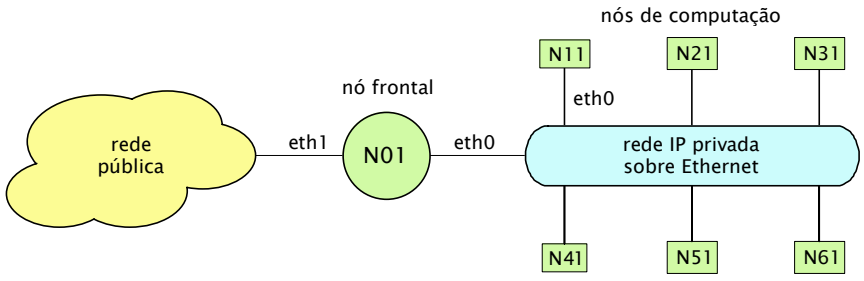

Figura 6.1 – Arquitectura Rocks aplicada ao cluster C1.

6.2 Plataforma *multi-cluster*

A plataforma *multi-cluster* de desenvolvimento é constituída por três *clusters* (C1, C2 e C3), com arquitectura semelhante à da Figura 6.1, instalados em ambiente VMware, cada um com os seguintes recursos:

- C1 = {Nós: N01, N11, N21; Rede IP Privada: 10.1.1.0/24}
- C2 = {Nós: N02, N12, N22, N32; Rede IP Privada: 10.2.2.0/24}
- C3 = {Nós: N03, N13, N23; Rede IP Privada: 10.3.3.0/24}

Nos três *clusters* o nó N0*i* representa o nó frontal, em que *i* corresponde ao índice do seu *cluster*. Para além de se encontrar ligado à rede privada do *cluster* também está ligado a uma rede IP comum, 192.168.10.0/24, que neste contexto de testes representa uma rede pública. Na Figura 6.2 representam-se os recursos que constituem a plataforma *multi-cluster* de desenvolvimento.

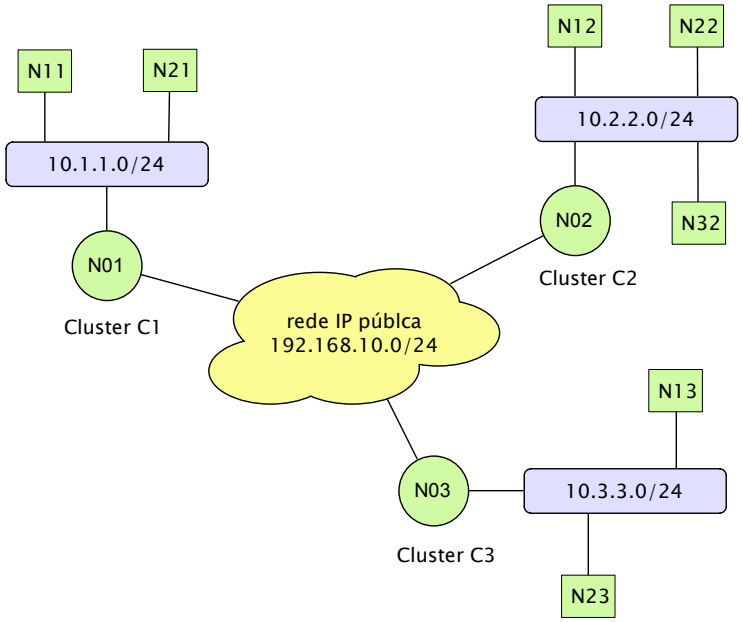

Figura 6.2 – Plataforma *multi-cluster* de desenvolvimento.

6.2.1 Rede de comunicações entre *clusters*

Como se pode constatar, estes três *clusters* partilham uma rede pública e os seus nós frontais podem comunicar entre si. No entanto, não é possível a comunicação entre as três redes privadas. Isto é, os nós de computação dos diferentes *clusters* não comunicam entre si. Por exemplo, o nó N11 de C1 não pode comunicar com qualquer nó de C2 e C3.

Isto acontece porque as redes privadas dos *clusters* "não se vêem" entre si. Os nós frontais N01, N02 e N03 funcionam como *gateways*, permitindo a comunicação da sua rede privada para o exterior, através do serviço de NAT. Contudo, não podem estender a comunicação para as redes privadas dos outros *clusters* porque desconhecem a forma de lá chegarem. Não dispõem de rotas de encaminhamento de tráfego para essas redes privadas. Como se pode ver na Figura 6.3, as tabelas de encaminhamentos dos nós frontais não têm nenhuma entrada para as redes privadas dos *clusters*: 10.1.1.0/24, 10.2.2.0/24 e 10.3.3.0/24. Apenas têm rotas para as respectivas redes privadas e para a rede pública.

Tabela de encaminhamento de N01 (C1)

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.1.0        *               255.255.255.0   U     0      0        0 eth0
192.168.100     *               255.255.255.0   U     0      0        0 eth1
169.254.0.0     *               255.255.0.0     U     0      0        0 eth1
224.0.0.0       *               240.0.0.0       U     0      0        0 eth0
```

Tabela de encaminhamento de N02 (C2)

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.2.2.0        *               255.255.255.0   U     0      0        0 eth0
192.168.100     *               255.255.255.0   U     0      0        0 eth1
169.254.0.0     *               255.255.0.0     U     0      0        0 eth1
224.0.0.0       *               240.0.0.0       U     0      0        0 eth0
```

Tabela de encaminhamento de N03 (C3)

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.3.3.0        *               255.255.255.0   U     0      0        0 eth0
192.168.100     *               255.255.255.0   U     0      0        0 eth1
169.254.0.0     *               255.255.0.0     U     0      0        0 eth1
224.0.0.0       *               240.0.0.0       U     0      0        0 eth0
```

Figura 6.3 – Tabelas de encaminhamento IP dos nós frontais.

Para que estes três *clusters* constituam um *multi-cluster* na perspectiva deste trabalho, é necessário que todos os nós de computação comuniquem entre si.

De acordo com a solução proposta no Capítulo 4, a comunicação entre as redes privadas dos *clusters* consegue-se estabelecendo redes privadas virtuais entre os respectivos nós frontais, usando para tal a tecnologia OpenVPN.

Pretende-se assim estabelecer um conjunto de ligações VPN que interligue os três nós frontais de forma que as suas redes privadas comuniquem entre si. O modelo deste conjunto de ligações obedece a uma lógica definida anteriormente, representada na Figura 4.4, baseada em dois princípios: 1) com a excepção do nó frontal do primeiro *cluster*, C1 neste caso, todos os nós frontais são simultaneamente servidores e clientes; 2) o nó frontal do último *cluster* a integrar a plataforma, supondo que há uma ordem, é servidor de VPN para todos os outros nós frontais. A aplicação desta lógica à plataforma de desenvolvimento está representada na Figura 6.4.

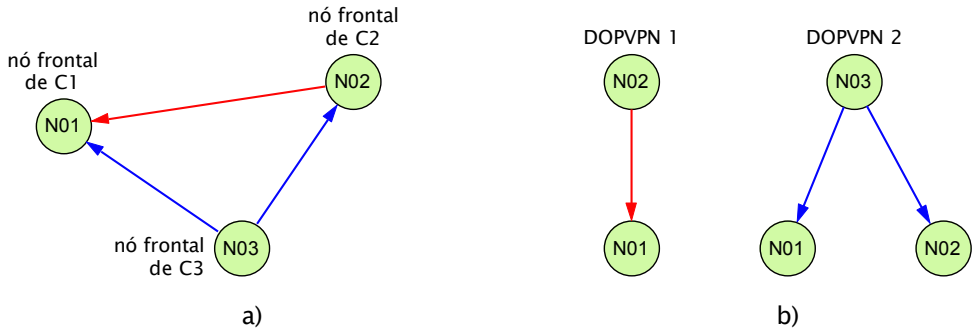

Figura 6.4 – a) Ligações VPN entre os três nós frontais e b) respectivos DOVPN.

A solução constrói-se definindo dois Domínios OpenVPN: DOVPN-1 em que N02 é servidor e N01 é cliente; DOVPN-2 em que desta vez N02 é cliente, assim como N01, e N03 é servidor de VPN.

Avançando para a configuração dos servidores e clientes OpenVPN importa destacar aspectos das configurações OpenVPN no que diz respeito a: 1) método de autenticação utilizado e 2) processo de encaminhamento de tráfego IP. De imediato apresenta-se a criação da PKI da plataforma *multi-cluster*, incluindo a emissão de chaves e certificados digitais. Depois é abordada a configuração dos Domínios OpenVPN.

Autenticação e segurança entre *clusters*

Como se referiu no Capítulo 3, o OpenVPN é utilizado em modo TLS, implicando autenticação baseada em certificados digitais X509. Para além da vantagem inerente à segurança garantida por este método de autenticação, o modo TLS permite ao servidor estabelecer múltiplas ligações com um único processo `openvpn`.

Para viabilizar este mecanismo de autenticação é necessário a criação de uma PKI (*Public Keys Infraestruture*), ou Infra-estrutura de Chaves Públicas, que permite a emissão e gestão dos certificados digitais para os nós servidores e clientes.

Construção da PKI do *multi-cluster*

Uma PKI em OpenPVN é elaborada com recurso ao pacote software livre OpenSSL. Não sendo um processo simples, é no entanto facilitado pelo conjunto de programas, `easy-rsa`, que o OpenVPN disponibiliza juntamente com as suas distribuições. Neste caso a PKI será criada no nó frontal N01 que concentrará a gestão relacionada com a plataforma *multi-cluster*.

A criação da PKI em N01 consiste nos seguintes passos:

1) Criação da CA (*Certificate Authority*) que implica ajustar parâmetros definidos no ficheiro `vars` necessários para gerar a chave DH (*Diffie-Hellman*) e o certificado da própria CA:

```
#parametros para a chave DH
Export KEY_SIZE=2048
#parametros para o certificado CA
```

```
Export KEY_CONTRY=PT
Export KEY_PROVINCE=N/A
Export KEY_CITY="Porto"
Export KEY_ORG="FEUP"
Export KEY_EMAIL="mrs04010@fe.up.pt"
```

De seguida assumir estes parâmetros,

```
$. ./vars
```

e limpar totalmente a PKI que ficará em: `opt/mc/openvpn/ca`,

```
$./clean-all
```

2) Gerar a chave *Diffie-Hellman* que cifrará a troca de chaves entre servidores e clientes, e gerar o certificado da CA.

```
$./build-dh
```

Este programa é equivalente ao comando:

```
$openssl dhparam -out opt/mc/openvpn/ca/dh2048.pem 2048
```

```
$./build-ca
```

Durante a criação do certificado de CA é solicitada a confirmação dos parâmetros editados em `vars`. A utilização de `build-ca` é equivalente à execução do comando:

```
$openssl req -days 3650 -nodes -new -x509 -keyout ca.key \
-out ca.crt -config openssl.cnf
```

o qual produz os ficheiros `ca.crt` e `ca.key`, respectivamente, o certificado e a chave privada de CA que assinará os certificados de servidores e clientes.

3) Gerar certificados de nós frontais enquanto servidores e clientes.

Para esta plataforma de desenvolvimento é necessário gerar certificados para os nós servidores OpenVPN N02 e N03, e para os nós clientes N01, N02 e N03. Repare-se que o nó N02 é simultaneamente servidor e cliente.

Assim, a geração de certificado do servidor N02 é realizada usando mais um programa do OpenVPN:

```
$./build-key-server N02-srv
```

Também aqui é solicitado a confirmação dos parâmetros do certificado. Como CN (*Common Name*) é indicado o nome do nó: `N02`.

Neste caso o programa `build-key-server` equivale à execução de dois comandos do OpenSSL. O primeiro gera a chave privada de `N02 e` um pedido de emissão de certificado à CA, e o segundo gera o certificado propriamente dito, assinado pela CA.

```
$openssl req -days 3650 -nodes -new -keyout N02-srv.key \
-out N02-srv.csr -extensions server –config openssl.cnf
```

```
$openssl ca -days 3650 -out N02-srv.crt -in N02-srv.csr \
-extensions server –config openssl.cnf
```

O certificado para o nó frontal N03 é gerado de forma idêntica. Apenas difere no parâmetro CN que é `N03`. No final ter-se-á o par `N03-srv.{crt, key}`.

Passa-se agora à geração dos certificados dos clientes OpenVPN, ou seja, dos nós N01 e N02. N03 não será cliente OpenVPN.

Geração de certificado do cliente N01:

```
$./build-key N01-clt
```

O parâmetro CN neste caso é N01. Todos os outros parâmetros se encontram definidos em `vars`. Mais uma vez o programa `build-key` corresponde à execução de um comando OpenSSL:

```
$openssl req -days 3650 -nodes –new -keyout N01-clt.key \
-out N01-clt.csr –config openssl.cnf

$openssl ca -days 3650 -out N01-clt.crt –in N01-clt.csr \
–config openssl.cnf
```

Nesta fase dispõe-se de uma PKI em N01 com os certificados dos nós frontais (servidores e clientes). A adição de mais *clusters* à plataforma implicará a emissão dos respectivos certificados como servidores e como clientes.

De imediato avança-se para a configuração dos domínios OpenVPN, já identificados anteriormente, no que diz respeito às componentes de autenticação e segurança.

Configuração de DOVPN-1

Este domínio é apenas constituído pelo cliente N01 e pelo servidor N02. Para cada um deles é necessário transferir para:

- N01 – a sua chave privada de cliente, `N01-clt.key`, respectivo certificado digital, `N01-clt.crt`, e o certificado digital de CA;
- N02 – a sua chave privada de servidor, `N02-srv.key`, respectivo certificado digital, `N02-srv.crt`, e o certificado digital de CA.

A configuração OpenVPN de ambos, relativa a autenticação e segurança, corresponde à definição das respectivas chaves privadas e certificados digitais, bem como, ao certificado digital da CA e chave DH que são comuns a todos os nós da plataforma. Especificam-se ainda directivas que diminuem os riscos inerentes a eventuais vulnerabilidades do código OpenVPN. As directivas `user` e `group` especificam que o processo `openvpn` corre como pertencente ao utilizador `nobody`, ou seja, com permissões reduzidas, e a directiva `chroot` restringe o acesso deste utilizador à directoria `/opt/mc/openvpn` no âmbito da execução do OpenVPN. Na Figura 6.5 e na Figura 6.6 apresentam-se as configurações parciais de N02 e N01.

Esta parte da configuração garante a autenticidade e a confidencialidade da comunicação entre os nós frontais de todos os *clusters*. Quando dois nós frontais se ligam entre si, cada um fornece ao outro o seu certificado digital. O nó que recebe o certificado verifica se este foi assinado pela Autoridade de Certificação da plataforma *multi-cluster*, especificada com a directiva `ca`. Caso as verificações de ambos os lados

sejam bem sucedidas, então a negociação TLS é concluída com êxito, ambos os nós trocam as chaves temporárias de sessão e o túnel IP fica apto para a comunicação de dados entre os dois nós frontais.

```
# Certificados CA e servidor N02, chave privada do servidor
cd /opt/mc/openvpn
ca ca.crt
cert N02-srv.crt
key N02-srv.key

# Parametros Diffie-Hellman
dh dh2048.pem

# Outros parametros de seguranca
user nobody
group nobody
chroot /opt/mc/openvpn
```

Figura 6.5 – Configuracao parcial do servidor OpenVPN N02.

```
# Certificados de CA e servidor, chave privada do servidor
cd /opt/mc/openvpn
ca ca.crt
cert N01-clt.crt
key N01-clt.key

# Outros parâmetros de segurança
user nobody
group nobody
chroot /opt/mc/openvpn
```

Figura 6.6 – Configuração parcial do cliente OpenVPN N01.

Configuração de DOVPN-2

A configuração deste domínio relativamente a autenticação é semelhante a DOVPN-1. Terá como servidor o nó frontal do *cluster* C3, N03, e como clientes os nós frontais dos *clusters* C1 e C2, respectivamente N01 e N02. O nó N01 já tem todos os seus dados e N02 já possui o certificado da CA, por isso é apenas preciso transferir para:

- N02 – a sua chave privada de cliente, `N02-clt.key`, e o respectivo certificado digital, `N02-clt.crt`.
- N03 – a sua chave privada de servidor, `N03-srv.key`, respectivo certificado digital, `N03-srv.crt`, e o certificado digital de CA.

No contexto deste domínio o nó N02 é cliente e por isso a sua configuração terá de incluir agora a sua chave e certificado de cliente, à semelhança de N01 que também é cliente. Quanto N03, só é servidor e só tem uma configuração de OpenVPN. Estas configurações são semelhantes às apresentadas nas Figuras Figura 6.5 e Figura 6.6.

Comunicação entre *clusters*

Esclarecidos os pormenores de configuração relacionados com a autenticação e segurança das ligações VPN, importa agora aferir como o OpenVPN implementa o encaminhamento para as redes privadas dos três *clusters* da plataforma *multi-cluster* e como se estabelecem as ligações VPN entre os nós frontais.

Encaminhamento de tráfego entre *clusters*

É uma missão do OpenVPN permitir a comunicação entre todos os nós dos vários *clusters*, de uma forma transparente e com a menor intervenção administrativa possível. Isto consegue-se ajustando as configurações do OpenVPN para anunciar rotas de encaminhamento das redes IP privadas dos *clusters*. É deste aspecto que a seguir se dá conta.

No caso desta plataforma existem 3 *clusters*, cada um com uma rede privada. Em cada *cluster* o nó frontal assegura a função de *gateway* que permite a comunicação da rede privada para o exterior, e só neste sentido, através do serviço de NAT que traduz todos os endereços IP da rede privada no endereço IP público do nó frontal. A comunicação do exterior para a rede privada não é possível.

Nas tabelas de encaminhamento já apresentadas na Figura 6.3 constata-se que cada nó frontal "desconhece" as redes privadas dos outros dois *clusters*, não existindo rotas de encaminhamento para essas redes. Nessas tabelas de encaminhamento a rede IP 192.168.10.0/24 representa a rede pública sobre a qual se estabelecerão as ligações OpenVPN.

As configurações de um servidor e de um cliente que estabelecem entre si uma ligação OpenVPN, devem incluir directivas que anunciem a rede privada de cada um dos lados da ligação.

Descrevem-se essas directivas para as configurações de DOVPN-1, em que o nó N01 é cliente do servidor OpenVPN instalado em N02. Analisam-se de seguida as directivas das configurações de ambos os nós que estão relacionadas com a comunicação IP entre esses dois nós. Para DOVPN-2 as directivas são semelhantes.

Directivas da configuração do servidor N02:

- dev tun

  Especifica a utilização da interface virtual de rede `tun`.

- server 172.16.2.0 255.255.255.0

  Cada servidor implementa o serviço DHCP, o qual fornece aos seus nós clientes o endereço IP das respectivas interfaces virtuais, do tipo `tun`. Estes endereços IP pertencem a uma rede IP de interligação definida pela directiva `server`. A interface `tun0` de N01 terá um endereço desta rede: 172.16.2.6.

- push "route 10.2.2.0 255.255.255.0"

  Esta directiva permite ao servidor N02 indicar a rede IP privada do seu *cluster* aos nós frontais seus clientes. Neste caso trata-se da rede 10.2.2.0/24. O cliente N01 recebe esta informação e adiciona à sua tabela de encaminhamento a rota

indicando que o tráfego para aquela rede deve ser enviado para o servidor N02, mais concretamente para o seu endereço 172.16.2.1 associado ao interface `tun0`. Ou seja, em N01, cliente ligado a N02, será adicionada a seguinte rota:

```
Destination Gateway    Genmask       Flags MSS Window irtt Iface
10.2.2.0    172.16.2.1 255.255.255.0 UG    0   0      0    tun0
```

- client-config-dir ccd

  ```
  route 10.1.1.0 255.255.255.0
  ```

  O conjunto destas duas instruções na configuração de N02 introduzem na sua tabela de encaminhamento de tráfego uma rota para a rede privada do *cluster* C1, acessível através de N01.
  `client-config-dir` especifica a directoria `ccd` onde se localizam ficheiros com a designação dos CN (nomes) dos clientes, ficheiros esses que contêm por sua vez a rede associada a cada cliente. Concretizando, N02 tem como cliente N01, assim em `ccd` é criado o ficheiro `N01` que contém a seguinte instrução:

  ```
  iroute 10.1.1.0 255.255.255.0
  ```

  A qual informa o servidor OpenVPN N02 que a rede 10.1.1.0/24 é a rede privada de C1, acessível através de N01. Esta associação é apenas interna ao OpenVPN. No entanto, N02 deve ter uma entrada na sua tabela de encaminhamento para aquela rede, o que será realizado pela directiva `route 10.1.1.0 255.255.255.0`. Assim, o tráfego gerado na rede privada de C2 com destino à rede privada de C1 é enviado para N02, que de acordo com a sua tabela de encaminhamento envia para o interface `tun0`, ou seja, para o processo `openvpn`. Este por sua vez, tem internamente associado a rede 10.1.1.0 ao cliente N01 e reencaminha o tráfego para este nó central através do túnel IP criado entre N01 e N02.

Directivas da configuração do cliente N01:

- dev tun

  ```
  proto udp
  remote 192.168.10.2 1194
  ```

  Estas são as directivas mais relevantes na configuração do cliente N01, em que `dev tun` especifica a utilização da interface virtual, enquanto `proto udp` determina a utilização do protocolo UDP na comunicação com o servidor, cujo endereço IP público, 192.168.10, e cujo porto UDP, 1194, são definidos pela directiva `remote`.

Depois de estabelecidos todos os túneis, os três *clusters* comunicam entre si: os nós de computação de um dos *clusters* deverão comunicar com os nós dos outros dois *clusters*. Para que isso aconteça é necessário que cada nó frontal (N01, N02 e N03) disponha de rotas para as redes privadas dos outros *clusters*, para além da rota para a rede privada do seu *cluster*.

Consultando com o comando `route` a tabela de encaminhamento de cada nó frontal, apresentadas na Figura 6.7, verifica-se que tal acontece.

Tabela de encaminhamento em N01 depois de OpenVPN

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.3.1      172.16.3.5      255.255.255.255 UGH   0      0        0 tun1
172.16.3.5      *               255.255.255.255 UH    0      0        0 tun1
172.16.2.1      172.16.2.5      255.255.255.255 UGH   0      0        0 tun0
172.16.2.5      *               255.255.255.255 UH    0      0        0 tun0
10.2.2.0        172.16.2.5      255.255.255.0   UG    0      0        0 tun0
10.10.10.0      *               255.255.255.0   U     0      0        0 eth1
10.1.1.0        *               255.255.255.0   U     0      0        0 eth0
10.3.3.0        172.16.3.5      255.255.255.0   UG    0      0        0 tun1
169.254.0.0     *               255.255.0.0     U     0      0        0 eth1
224.0.0.0       *               240.0.0.0       U     0      0        0 eth0
```

Tabela de encaminhamento em N02 depois de OpenVPN

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.3.1      172.16.3.9      255.255.255.255 UGH   0      0        0 tun1
172.16.2.2      *               255.255.255.255 UH    0      0        0 tun0
172.16.3.9      *               255.255.255.255 UH    0      0        0 tun1
10.2.2.0        *               255.255.255.0   U     0      0        0 eth0
10.10.10.0      *               255.255.255.0   U     0      0        0 eth1
10.1.1.0        172.16.2.2      255.255.255.0   UG    0      0        0 tun0
172.16.2.0      172.16.2.2      255.255.255.0   UG    0      0        0 tun0
10.3.3.0        172.16.3.9      255.255.255.0   UG    0      0        0 tun1
169.254.0.0     *               255.255.0.0     U     0      0        0 eth1
224.0.0.0       *               240.0.0.0       U     0      0        0 eth0
```

Tabela de encaminhamento em N03 depois de OpenVPN

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.3.2      *               255.255.255.255 UH    0      0        0 tun0
10.2.2.0        172.16.3.2      255.255.255.0   UG    0      0        0 tun0
10.10.10.0      *               255.255.255.0   U     0      0        0 eth1
10.1.1.0        172.16.3.2      255.255.255.0   UG    0      0        0 tun0
172.16.3.0      172.16.3.2      255.255.255.0   UG    0      0        0 tun0
10.3.3.0        *               255.255.255.0   U     0      0        0 eth0
169.254.0.0     *               255.255.0.0     U     0      0        0 eth1
224.0.0.0       *               240.0.0.0       U     0      0        0 eth0
```

Figura 6.7 – Tabelas de encaminhamento IP depois das ligações OpenVPN activadas.

Constata-se que em cada nó frontal passa a existir rotas de encaminhamento para as redes privadas dos *clusters* que integram a plataforma *multi-cluster*. Concretamente:

- Em N01: rota para a rede 10.2.2.0 de C2; rota para a rede 10.3.3.0 de C3;
- Em N02: rota para a rede 10.1.1.0 de C1; rota para a rede 10.3.3.0 de C3;
- Em N03: rota para a rede 10.1.1.0 de C1; rota para a rede 10.2.2.0 de C2.

Cada nó frontal actua também como encaminhador de tráfego IP entre a rede privada do seu *cluster* e as redes privadas dos outros *clusters*.

Da leitura da Figura 6.7 também se pode constatar que o nó frontal N03 só possui uma interface do tipo `tun` enquanto os outros dois nós têm duas interfaces: `tun0` e

`tun1`. Tal se deve ao facto de N03 só ter um processo `openvpn`, porque só é servidor, e N01 e N02 terem dois processos. N01 porque é duas vezes cliente, N02 porque é uma vez servidor e outra cliente. Estes factos justificam também o menor número de rotas (8) em N03 que em N01 e N02 (10 rotas).

Estabelecimento das ligações

Efectuadas as configurações dos servidores e clientes OpenVPN é possível estabelecer as ligações entre os três *clusters* C1, C2 e C3.

Primeiro é necessário activar o servidor em cada um dos domínios OpenVPN, e depois activar os respectivos clientes.

Em DOVPN-1,

1) Levantar serviço OpenVPN em N02:

```
$ openvpn --config /etc/openvpn/n02-srv.conf –daemon
```

2) Activar o cliente OpenVNP em N01:

```
$ openvpn --config /etc/openvpn/n01-n02.conf –daemon
```

A execução deste cliente, com a configuração definida em `n01-n02.conf`, estabelece a ligação VPN entre N01 e N02.

Em DOVPN-2,

3) Levantar serviço OpenVPN em N03:

```
$ openvpn --config /etc/openvpn/n03-srv.conf –daemon
```

4) Activar o cliente OpenVNP em N01:

```
$ openvpn --config /etc/openvpn/n01-n03.conf –daemon
```

5) Activar o cliente OpenVNP em N02:

```
$ openvpn --config /etc/openvpn/n02-n03.conf –daemon
```

Com estes três passos estabelece-se uma ligação entre o cliente N01 e o servidor N03 e outra ligação entre N02 e N03. Reafirma-se que em N01 passam a existir dois processos `openvpn`, ambos clientes, e em N02 também, mas um processo cliente enquanto o outro é um processo servidor.

A Figura 6.8 mostra os túneis IP que se estabelecem entre os três nós frontais depois de activados os processos `openvpn`:

- Túnel N02 (`tun0`-172.16.2.2) ↔ N01 (`tun0`-172.16.2.5);
- Túnel N03 (`tun0`-172.16.3.2) ↔ N01 (`tun1`-172.16.3.5);
- Túnel N03 (`tun0`-172.16.3.2) ↔ N02 (`tun1`-172.16.3.9).

N03 estabelece dois túneis sobre a mesma interface virtual `tun0` enquanto N01 e N02 estabelecem dois túneis através de interfaces diferentes, `tun0` e `tun1`.

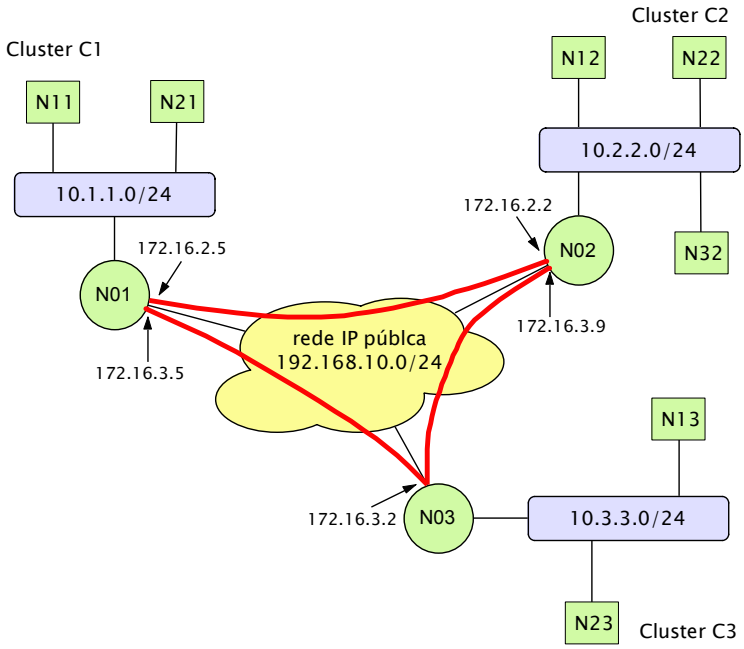

Figura 6.8 – Túneis IP estabelecidos entre os três nós frontais.

Configurações totais dos Domínios OpenVPN

A terminar a exposição sobre a configuração da rede de interligação do *multi-cluster* de desenvolvimento, apresentam-se as configurações globais de ambos os Domínios OpenVPN.

DOVPN-1 é constituído pelos nós frontais N01 e N02, respectivamente, cliente e servidor. Os dados necessários para a sua configuração:

- Certificado digital da CA: `ca.crt`;
- Chave DH: `dh2048.pem`;
- Chave privada e certificado do cliente N01: `N01-ctl.key` e `N01-ctl.crt`;
- Chave privada e certificado do servidor N02: `N02-srv.key` e `N02-srv.crt`;
- Endereço IP público do servidor N02: 192.168.10.2;
- Rede IP de interligação do domínio: 172.16.2.0/24;
- Rede privada de C2, *cluster* servidor: 10.2.2.0/24;
- Rede privada de C1, *cluster* cliente: 10.1.1.0/24.

Na Figura 6.9 apresentam-se as configurações integrais do servidor N02 e do cliente N01. A negrito encontram-se as directivas que particularizam ambas as configurações.

n02-srv.conf

```
dev tun
proto udp
port 1194
cd /opt/mc/openvpn
ca ca.crt
cert N02-srv.crt
key N02-srv.key
dh dh1024.pem
server 172.16.2.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "route 10.2.2.0 255.255.255.0"
client-config-dir ccd
route 10.1.1.0 255.255.255.0
keepalive 10 120
duplicate-cn
user nobody
group nobody
chroot /opt/mc/openvpn
tun-mtu 1500
mssfix 1400
persist-key
persist-tun
verb 3
```

n01-n02.conf

```
client
dev tun
proto udp
remote 192.168.10.2 1194
resolv-retry infinite
nobind
cd /opt/mc/openvpn
ca ca.crt
cert N01-ctl.crt
key N01-ctl.key
user nobody
group nobody
chroot /opt/mc/openvpn
tun-mtu 1500
mssfix 1400
persist-key
persist-tun
verb 3
```

Figura 6.9 – Configurações OpenVPN de N01 e N02 em DOPVN-1.

DOVPN-2 é constituído pelos nós frontais N01 e N02, como clientes, e por N03 como servidor. Os dados necessários para a sua configuração:

- Certificado digital da CA: `ca.crt`;
- Chave DH: `dh2048.pem`;
- Chave privada e certificado do cliente N01: `N01-ctl.key` e `N01-ctl.crt`;
- Chave privada e certificado do cliente N02: `N02-ctl.key` e `N02-ctl.crt`;
- Chave privada e certificado do servidor N03: `N03-srv.key` e `N03-srv.crt`;
- Endereço IP público do servidor N03: 192.168.10.3
- Rede IP de interligação do domínio DOVPN-2: 172.16.3.0/24;
- Rede privada de C3, *cluster* servidor: 10.3.3.0/24;
- Redes privadas de C1 e C2, *clusters* clientes: 10.1.1.0/24 e 10.2.2.0/24.

Na Figura 6.10 encontram-se as configurações do servidor N03 e do cliente N02. Comparando as configurações dos servidores dos dois domínios OpenVPN, verifica-se que entre elas mudam os parâmetros das directivas de autenticação (`cert` e `key`), a directiva da rede de interligação do domínio (`server`) e as directivas que definem o encaminhamento no domínio (`push` e `route`). Repara-se que N03 tem a directiva duplicada, cada uma delas correspondendo aos *clusters* C1 e C2 que a si se ligam através dos nós frontais N01 e N02, respectivamente. Já da comparação das configurações dos clientes, se conclui que apenas se altera a directiva `remote` que especifica o servidor e as directivas `cert` e `key`.

n03-srv.conf

```
dev tun
proto udp
port 1194
cd /opt/mc/openvpn
ca ca.crt
cert N03-srv.crt
key N03-srv.key
dh dh1024.pem
server 172.16.3.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "route 10.3.3.0 255.255.255.0"
client-config-dir ccd
route 10.1.1.0 255.255.255.0
route 10.2.2.0 255.255.255.0
keepalive 10 120
duplicate-cn
user nobody
group nobody
chroot /opt/mc/openvpn
tun-mtu 1500
mssfix 1400
persist-key
persist-tun
verb 3
```

n02-n03.conf

```
client
dev tun
proto udp
remote 192.168.10.3 1194
resolv-retry infinite
nobind
cd /opt/mc/openvpn
ca ca.crt
cert N02-ctl.crt
key N02-ctl.key
user nobody
group nobody
chroot /opt/mc/openvpn
tun-mtu 1500
mssfix 1400
persist-key
persist-tun
verb 3
```

Figura 6.10 – Configurações OpenVPN de N03 e N02 em DOPVN-2.

Estas configurações também demonstram que há uma configuração tipo para servidores e outra configuração tipo para clientes. Criando um modelo para cada tipo, elaboram-se todas as configurações de um domínio OpenVPN aplicando os seus dados particulares a esses modelos.

6.3 Sistema de ficheiros *multi-cluster*

A criação de um Sistema de Ficheiros Alargado (SFA) a todo o *multi-cluster* é a próxima etapa. Essencialmente pretende-se implementar um serviço de NFS comum a todos os *clusters*. Como se explanou no Capítulo 4 a implementação do SFA implica as seguintes tarefas em cada *cluster*: 1) criação do utilizador `sfa`, 2) criação de directorias de trabalho *paralelas*, 3) instalação de procuradores NFS de níveis 1 e 2, 4) activação do sistema de *cache*. Seguindo este roteiro procura-se demonstrar a construção do SFA.

6.3.1 Criação do utilizador `sfa`

A existência deste utilizador justifica-se pela necessidade de partilhar uma parte do sistema de ficheiros de um *cluster* com os restantes *clusters*. Essa parte é representada pela directoria de trabalho do utilizador `sfa`, ou seja, `/export/home/sfa`.

A criação de `sfa` é feita da mesma forma que para um utilizador normal. Apenas por questão de conveniência, mas não obrigatória, deve ter a mesma identidade,

`uid/gid`, em todos os *clusters*. No *multi-cluster* de desenvolvimento convencionou-se que essa identidade é: `uid`=600 e `gid`=600.

Assim, a adição de `sfa` a *cluster* corresponde à execução do comando:

```
# adduser –u 600 –g 600 –c "Sistema de Ficheiros Alargado" sfa
```

A execução é realizada no nó frontal de *cluster* (N01, N02 e N03), após a qual o novo utilizador é adicionado a todos os nós de computação através do serviço 411 do Rocks. No entanto, a sua existência só interessa nos nós frontais.

A directoria `/export/home/sfa` é virtualmente repartida pelos utilizadores do SFA, tendo cada um a *fatia* correspondente.

6.3.2 Criação de directorias de trabalho *paralelas*

Como é referido no Capítulo 4 é necessário definir em todos os nós uma estrutura de directorias de trabalho que é *paralela* há estrutura já existente em cada *cluster*. Estas directorias são os pontos de ligação[1] para a directoria `/export/home/sfa` exportada por NFS por cada nó frontal.

Para sustentar esta simulação é definido em cada um *cluster* uma conta de acesso pertencente ao mesmo utilizador *real* José Silva.

As contas de acessos são geridas por domínios de administração diferentes e por essa razão têm identidades diferentes: nome e `uid/gid`. A Tabela 6.1 resume esta situação no *multi-cluster*.

Tabela 6.1 – Contas de acesso de um utilizador dos 3 *clusters*.

Cluster	Utilizador	Conta de acesso	Directoria de trabalho	uid	gid
C1	José Silva	jsilva	/home/jsilva	520	510
C2	José Silva	silva	/home/silva	575	535
C3	José Silva	js	/home/js	521	512

José Silva é um utilizador comum aos três *clusters* e por isso um potencial utilizador do SFA. Um utilizador apenas com conta de acesso num *cluster* não usará o SFA, uma vez que não utiliza os outros dois *clusters*, isto é, não é um utilizador do *multi-cluster*.

As directorias de trabalho de José Silva nos nós de computação de cada *cluster* são criadas com o utilitário `cluster-fork` disponibilizado pelo Rocks, o qual executa automaticamente em cada nó de computação, a partir do nó frontal, um ou mais comandos passados como parâmetros. Segue a sua execução nos três *clusters*.

[1] Tradução de *mount points*.

Em N01 de C1:

```
# cluster-fork 'mkdir /sfa/jsilva & chown /sfa/jsilva 520 &
\ chgrp /sfa/jsilva 510'
```

Em N02 de C2:

```
# cluster-fork 'mkdir /sfa/silva & chown /sfa/silva 575 & \
chgrp /sfa/silva 535'
```

Em N03 de C3:

```
# cluster-fork 'mkdir /sfa/js & chown /sfa/js 521 & \
  chgrp /sfa/js 512'
```

A Tabela 6.2 reflecte já as novas directorias do utilizador no âmbito do SFA, tendo agora em cada *cluster* duas directorias de trabalhos, mas ambas com igual par UID/GID. A primeira directoria diz respeito ao sistema de ficheiros local e a segunda insere-se em SFA.

Tabela 6.2 – Directorias de trabalho *paralelas* de um utilizador.

Cluster	Utilizador	Conta de acesso	Directoria de trabalho	uid	gid
C1	José Silva	jsilva	/home/jsilva /sfa/jsilva	520	510
C2	José Silva	silva	/home/silva /sfa/silva	575	535
C3	José Silva	js	/home/js /sfa/js	521	512

Como já se referiu, o desafio que se coloca é permitir ao utilizador e às suas aplicações o acesso de forma transparente aos seus ficheiros armazenados nos diferentes *clusters*. Implementar esta facilidade sem alterações ao nível do ambiente de execução passa pela instalação de procuradores NFS, que terão como principal função o mapeamento das diversas identidades que o utilizador assume em cada um dos *clusters*.

6.3.3 Instalação de procuradores NFS

O procurador NFS é constituído por dois processos: `pvfs.mountd` e `pvfs.nfsd`. Actuando em conjunto, ambos têm como funções: converter a identidade do utilizador, autenticar os pedidos de acesso à partilha especificada pelo utilizador e, em caso de sucesso, encaminhar os pedidos para o servidor NFS nativo ou para outro procurador NFS.

No SFA são usados dois níveis de procuradores: procuradores de nível 1 (*pnfsn1*) e procuradores de nível 2 (*pnfsn2*). Ambos os tipos de procuradores são instalados nos nós frontais dos *clusters*: N01, N02 e N03.

Procuradores NFS de nível 1

Estes procuradores são permanentes e interagem com o serviço de NFS nativo. A sua activação implica duas acções: 1) Proceder a pequenas alterações na configuração do NFS nativo 2) Configurar e activar *pnfsn1* no arranque dos nós frontais.

Descrevem-se a seguir as alterações ao NFS nativo.

O serviço de NFS é composto pelo processo `nfsd`, o qual utiliza o porto 2049 UDP/TCP, e pelo processo `rcp.mountd` que utiliza portos UDP/TCP atribuídos por um dos serviços RPC: `portmap`.

O processo `pvfs.nfsd` comunica com o processo `nfsd` através do porto 2049. A comunicação entre o processo `pvfs.mountd` e o processo `rpc.mountd` implica a fixação dos seus portos UDP/TCP. Para isso edita-se o ficheiro `/etc/init.d/nfs` e especifica-se o porto 33333 para `rpc.mountd`:

```
daemon rpc.mountd –p 33333 –o 4096 $RPCMOUNTDOPTS
```

Para além da predefinição do porto, incrementou-se para o máximo (4096) o número de descritores de ficheiros.

Como se disse, *pnfsn1* interage com o NFS nativo, acedendo à partição `/export` exportada para os nós do *cluster*. É agora necessário autorizar o acesso a partir do próprio nó frontal onde corre o serviço NFS, ou seja, autorizar o acesso a `localhost`, editando `/etc/exports`:

```
/export  10.1.1.0/255.255.255.0(rw)  localhost(rw)
```

De imediato procede à configuração de *pnfsn1* e à sua activação.

A configuração de *pnfsn1* é realizada em três ficheiros: `n1_exports_mountd`, `n1_exports_nfsd` e `n1.map`. Cada um deles com o seguinte conteúdo:

- /etc/pvfs/n1_exports_mountd

```
/export/home/sfa localhost(rw) 10.1.1.0/255.255.255.0(rw)\
172.16.0.0/255.255.0.0(rw)
```

- /etc/pvfs/n1_exports_nfsd

```
/export/home/sfa localhost (rw,map_static=/etc/pvfs/n1_map)\
10.1.1.0/255.255.255.0 (rw,map_static=/etc/pvfs/n1_map)\
172.16.0.0/255.255.0.0(rw,map_static=/etc/pvfs/n1_map)
```

- /etc/pvfs/n1_map

```
uid   600 600
gid   600 600
```

Nestes ficheiros de configurações autoriza-se quem pode aceder à directoria `/export/home/sfa`, todos os nós do *cluster*, incluindo o próprio *localhost*, e todos os nós frontais do *multi-cluster*, os quais utilizam as ligações OpenVPN e por isso são identificados pelos seus endereços da rede de interligação 172.16.0.0. Também se

define qual o tipo de acesso, que no presente caso é de leitura e escrita (rw). Por último, define-se o mapeamento feito pelo procurador. Como se sabe, *pnfsn1* não tem esta função, razão pela qual as duas colunas do ficheio `/etc/pvfs/n1_map` são iguais.

Realizada a configuração é altura de proceder à sua activação. Para isso elabora-se a *script* `pnfsn1.sh` com o seguinte conteúdo:

```
echo "Iniciar PVFS.mountd"
/opt/mc/bin/pvfs.mountd -f /etc/pvfs/n1_exports_mountd -V 2\
-P 20001 -O 33333 -t udp -H localhost
echo "iniciar PVFS.nfsd"
/opt/mc/bin/pvfs.nfsd -f /etc/pvfs/n1_exports_nfsd -V 2 -P\
20002 -O 2049 -t udp -H localhost
```

Para activar *pnfsn1* no arranque dos nós frontais, edita-se de novo o ficheiro `/etc/init.d/nfs` no final do qual é acrescentada a *script* acima definida.

Fazendo `/etc/init.d/nfs restart` reinicia-se todos os processos de NFS e, consequentemente, activa-se *pnfsn1*. O processo `pvfs.mountd` estabelece ligação com `mountd` através do porto 33333 e "escuta" ligações no porto 20001, enquanto `pvfs.nfsd` comunica com o porto 2049 de `nfsd` e aceita conexões no porto 20002.

Estes procedimentos são realizados em N01. O mesmo processo se repete em N02 e N03, tendo em consideração o ajuste a fazer na configuração de cada *pnfsn1*.

Procuradores NFS de nível 2

Enquanto o procurador de nível 1 recebe pedidos de um procurador de nível 2 (*pnfsn2*) e os reencaminha para o servidor de NFS, o procurador de nível 2 recebe pedidos do cliente NFS (`mount`) e reencaminha esses pedidos para o procurador de nível 1.

A configuração de um *pnfsn2* é feita em função do utilizador, isto é, sendo cada *pnfsn2* dedicado a um determinado utilizador a sua configuração é feita tendo em conta o contexto desse utilizador. A implementação do SFA para um utilizador implica a execução, no nó frontal de cada *cluster* envolvido, de um *pnfsn2* em exclusivo para esse utilizador.

No *multi-cluster* de ensaio desenvolve-se agora o SFA para o utilizador José Silva já atrás caracterizado. Usando N02 como servidor de ficheiros, a partilha que é montada por todos os nós de computação é `/export/home/sfa/silva`. Haverá três *pnfsn2* dedicados a José Silva designados por: `pnfsn2.jsilva` (em N01), `pnfsn2.silva` (em N02) e `pnfsn2.js` (em N03). Para cada um destes procuradores é necessário preparar as respectivas configurações.

Exemplificando, a configuração de `pnfsn2.jsilva` (procurador NFS de nível 2, no *cluster* C1, para o utilizador `jsilva`) é constituída pelos seguintes ficheiros e respectivos conteúdos:

- /etc/pvfs/n2_exports_mountd.jsilva

```
/export/home/sfa/silva localhost(rw) \
10.1.1.0/255.255.255.0(rw)
```

- /etc/pvfs/n2_exports_nfsd.jsilva

```
/export/home/sfa/silva \
localhost(rw,map_static=/etc/pvfs/n2_map.jsilva) \
10.1.1.1/255.255.255.0(rw,map_static=/etc/pvfs/n2_map.jsilva)
```

- /etc/pvfs/n2_map.jsilva

```
uid  520 600
gig  510 600
```

Esta configuração mapeia a directoria `/sfa/jsilva`, em qualquer nó de C1, para a directoria `/export/home/sfa/silva`, exportada por N02 de C2.

O sufixo "jsilva" em todos os ficheiros indica que se trata da configuração do utilizador `jsilva` tendo C1 como cliente e C2 como servidor NFS.

Realizadas as configurações de `pnfsn2.jsilva` é possível activar o procurador usando a *script* `pnfsn2.jsilva.sh`:

```
#!/bin/sh
echo "A iniciar PVFS.mountd de jsilva em C1"
/opt/mc/bin/pvfs.mountd -f \
/etc/pvfs/n2_exports_mountd.jsilva -V 2 -P 30001 -O 20001 \
-t udp -H N02-pri
echo "A iniciar PVFS.nfsd de jsilva em C1"
/opt/mc/bin/pvfs.nfsd -f /etc/pvfs/n2_exports_nfsd.jsilva \
-V 2 -P 30002 -O 20002 -t udp -H N02-pri -A -D -T 10 \
-c /sfa/cache -i 1000
```

Relembra-se que cada nó frontal dispõe de dois acessos de rede: um para a rede privada do seu *cluster* (N02-pri), o outro para a rede pública (N02-pub). A *script* apresentada em cima usa N02-pri, registado no DNS com o endereço 10.2.2.1, rede privada de C2. Desta forma encaminha-se este tráfego pelo túnel IP entre N01 e N02, criado pelo OpenVPN, sendo N01 autenticado por *pnfsn1* de N02 com base no endereço IP da sua interface `tun0` (172.16.2.5).

Outro aspecto importante presente ainda na *script* em cima refere-se à activação da função *cache* de disco, a qual é parametrizada pelas opções `A`, `D`, `T=10`, `c=/sfa/cache` e `i=1000`, em que:

- `A` – activa *cache* atributos;
- `D` – activa *cache* de dados;
- `T=10` – especifica intervalo de sincronismo (segundos) de *cache* com o servidor;
- `c=/sfa/cache` – especifica directoria base da *cache*;
- `i=1000` – especifica *id* de sessão de *cache*.

Tendo o utilizador José Silva conta de acesso nos outros dois *clusters* (`silva` em C2 e `js` em C3), procede-se de forma semelhante à instalação dos respectivos procuradores de nível 2, realizando apenas as necessárias alterações nos ficheiros de configuração.

Executando a *script* `pnfsn2.silva.sh` em N02 e a *script* `pnfsn2.js.sh` em N03, activam-se todos os procuradores de nível 2 para o utilizador José Silva.

Finalmente, é altura de activar os clientes NFS através do comando `mount` que é executado em todos os nós de computação do *multi-cluster*, usando o comando `cluster-fork` nos nós frontais.

Em N01 de C1 executa-se:

```
# cluster-fork mount -t nfs -o port=30002,mountport=30001,\
vers=2,mountvers=2,udp,rsize=8192,wsize=8192  \
N01-pri:/export/home/sfa/silva /sfa/jsilva
```

Enquanto em N02 de C2 o comando toma a seguinte forma:

```
# cluster-fork mount -t nfs -o port=30002,mountport=30001,\
vers=2,mountvers=2,udp,rsize=8192,wsize=8192  \
N02-pri:/export/home/sfa/silva /sfa/silva
```

Por último a activação do cliente NFS em N03 de C3 corresponde a executar:

```
# cluster-fork mount -t nfs -o port=30002,mountport=30001,\
vers=2,mountvers=2,udp,rsize=8192,wsize=8192  \
N03-pri:/export/home/sfa/silva /sfa/js
```

Note-se que os parâmetros dos comandos anteriores são praticamente iguais. Apenas se altera o servidor ao qual os clientes NFS se ligam (N01-pri, N02-pri, N03-pri) e a directoria ponto de ligação (`/sfa/jsilva`, `/sfa/silva`, `/sfa/js`). Embora o servidor de ficheiros do utilizador José Silva seja remoto em C2, a ligação dos clientes NFS de C1 e C3 é feita ao respectivo nó frontal onde funciona a *cache* de disco. Verifica-se também através destes comandos que é utilizada *cache* de memória nos clientes.

Nesta fase o utilizador José Silva tem um sistema de ficheiros uno em todo o *multi-cluster*, podendo aceder de igual forma a partir de qualquer nó de computação. Assim, a simples experiência de listar ficheiros nos três *clusters* tem o resultado reproduzido em baixo.

Execução do comando `ls` em N11 de C1:

```
$ ls –la teste*
-rw-rw-r--   1 jsilva jsilva 294676 May 18 12:23 sfa.txt
-rw-r--r--   1 jsilva jsilva   7837 Mar 14 10:45 pvm.cfg
```

Execução do comando `ls` em N12 de C2:

```
$ ls –la
-rw-rw-r--   1  silva  silva 294676 May 18 12:23 sfa.txt
-rw-r--r--   1  silva  silva   7837 Mar 14 10:45  pvm.cfg
```

Execução do comando `ls` em N23 de C3:

```
$ ls –la
-rw-rw-r--   1     js     js 294676 May 18 12:23 sfa.txt
-rw-r--r--   1     js     js   7837 Mar 14 10:45 pvm.cfg
```

Como se pode constatar, sendo os mesmos ficheiros, apenas muda a identidade do respectivo proprietário. Na realidade os ficheiros estão armazenados em `/export/home/sfa/silva` de N02 e pertencem ao utilizador `sfa` como se pode ver através do seguinte comando em N02:

```
# ls –la /export/home/sfa/silva
-rw-rw-r--   1    sfa    sfa 294676 May 18 12:23 sfa.txt
-rw-r--r--   1    sfa    sfa   7837 Mar 14 10:45 pvm.cfg
```

Os procuradores de nível 2 fazem a conversão de identidades dando a cada utilizador final (*jsilva*, *silva* e *js*) privilégios de verdadeiros proprietários dos ficheiros.

Neste momento a plataforma *multi-cluster* de desenvolvimento consiste em três *clusters* com conectividade IP entre todos os nós de computação e dispõe de um sistema ficheiros NFS, alargado, fornecendo aos utilizadores com acesso a mais de um *cluster* a opção de terem um sistema de ficheiros uno.

Dispomos então de uma infra-estrutura computacional sobre a qual é possível implementar o Serviço de *Cluster* Virtual.

6.3.4 Desempenho de SFA

Sendo o Sistema de Ficheiros Alargado constituído por vários componentes pretende-se nesta secção apresentar o impacto qualitativo que cada um deles tem seu desempenho global.

Neste sentido, foi efectuada uma avaliação de desempenho do SFA do *multi-cluster* de desenvolvimento, em função dos seus componentes: NFS, OpenVPN e procuradores NFS, com e sem *cache*.

Seria adequado a utilização de ferramentas específicas como IOzone [47] e Bonnie [48], no entanto os resultados ambíguos de experiências preliminares demonstraram que estas ferramentas não se ajustam a sistemas de ficheiros de máquinas virtuais, como são as que constituem os nós do *multi-cluster* de desenvolvimento. Assim, optou-se por realizar esta avaliação de desempenho de uma forma simples, baseada na transferência de dados entre nós.

Concretamente, a avaliação de desempenho de SFA consistiu na realização de transferências sucessivas de um ficheiro de 100 MB (`teste.tar`), usando o comando *copy* do Linux, em quatro cenários diferentes, representados na Figura 6.11. Em cada um dos cenários foram efectuadas 10 transferências de dados.

O primeiro cenário, Figura 6.11 – a), consiste em realizar a transferência de dados no contexto do *cluster* C2, isto é; os dados são transferidos do nó frontal N02 para o nó de computação N12. Este caso é o mais simples, estando a apenas envolvido na transferência o sistema de ficheiros NFS do *cluster* C2. Os resultados obtidos neste cenário servem de referência para os cenários seguintes que envolvem outros componentes.

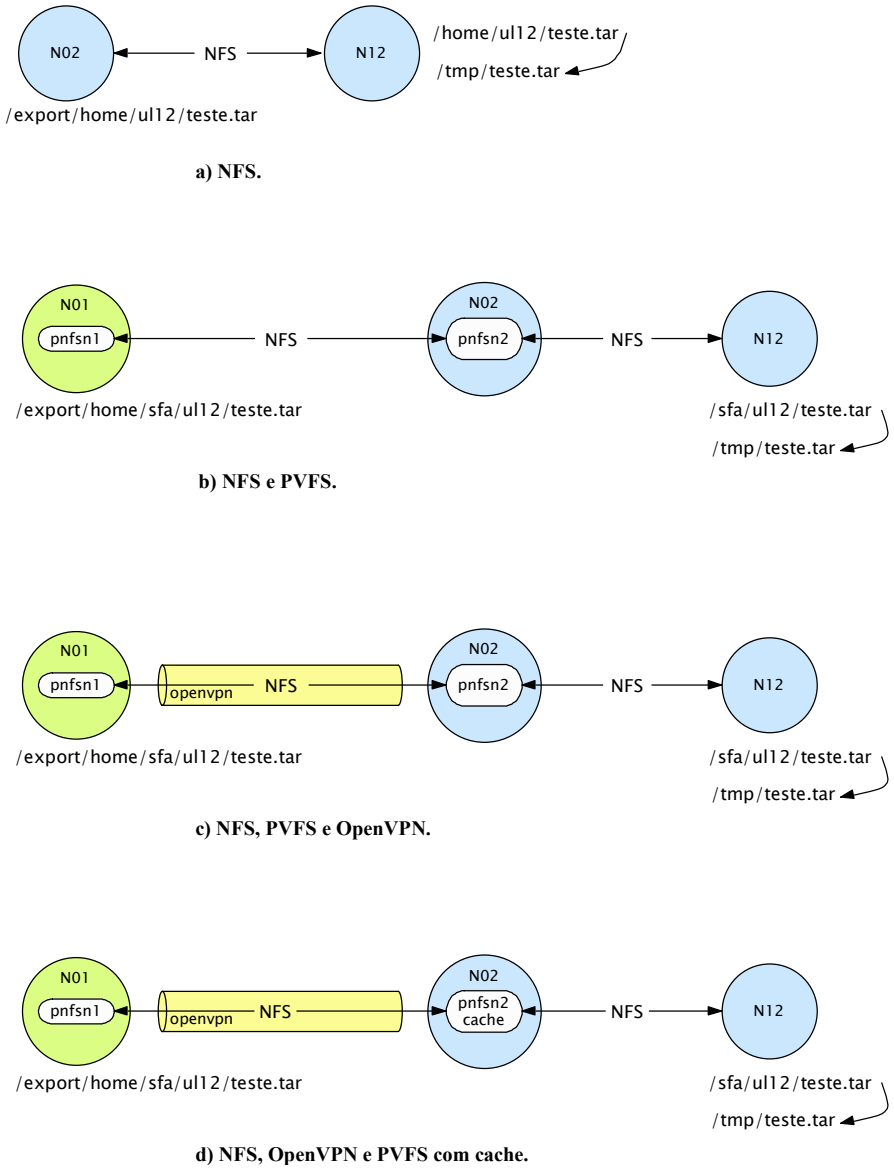

Figura 6.11 – Cenários de avaliação do desempenho de SFA.

Nos restantes cenários a transferência realiza-se entre nós de diferentes *clusters*, ou melhor, no contexto de um CV que integra C1 e C2, do nó frontal N01 para o nó de computação N12.

No segundo cenário, Figura 6.11 – b), são introduzidos os procuradores NFS de níveis 1 e 2 sem *cache*. Conforme consta na Tabela 6.4, os tempos de transferência medidos aumentam substancialmente, demonstrando que o processamento realizado por estes componentes tem um impacto considerável no desempenho de SFA, traduzido num tempo de transferência 5 vezes maior.

Tabela 6.3 – Resultados das transferências de dados nos 4 cenários.

Componentes SFA	NFS (N02→N12)	PNFS (N01→N12)	PNFS e OPVN (N01→N12)	PNFS c/ cache e OPVN (N01→N12)
T1	7s	37	79	40
T2	5s	30	67	14
T3	7s	28	66	12
T4	7s	29	66	12
T5	6s	30	65	13
T6	5s	28	70	14
T7	5s	29	73	13
T8	7s	28	72	13
T9	6s	28	69	14
T10	6s	29	67	13
Média	**6,1s**	29,6	69,4	**15,8**

Em c) da Figura 6.11 representa-se o terceiro cenário usado e em que é activado um túnel OpenVPN entre os nós frontais N01 e N02. Também aqui é bem visível a sobrecarga imposta pelo túnel fazendo com que a cópia de informação demore cerca de 10 vezes mais que a cópia efectuada no âmbito de um único *cluster*.

Por último, realiza-se a experiência com a solução final, em que os procuradores de NFS de nível 2 são configurados com *cache*, situação retratada em d) da Figura 6.11. Deste modo consegue-se melhorar substancialmente o desempenho de SFA, viabilizando tempos de transferência muito semelhantes aos do primeiro cenário – *cluster* físico.

Conforme se verifica na, a primeira transferência no quarto cenário é a mais demorada por corresponder à construção da *cache* em N02. As restantes 9 transferências beneficiam já da *cache*, uma vez que equivalem as transferências no âmbito do *cluster* C2, ou seja, de N02 para N12. Por essa razão os seus tempos são reduzidos.

A Figura 6.12 epresenta um gráfico que sintetiza os resultados e evidencia a aproximação entre o desempenho de um sistema de ficheiros de *cluster* baseado em NFS (primeira linha) e o SFA implementado com todos os seus componentes (segunda linha). No gráfico é bem visível a sobrecarga introduzida pelo OpenVPN, assim como, a importância da *cache* do procurador NFS de nível 2 no desempenho global de SFA.

Esta análise do comportamento de SFA não foi exaustiva. Avaliou o seu desempenho usando operações de leitura de blocos de dados, remotos num nó frontal, realizadas em qualquer nó do *multi-cluster*. A escrita remota de dados seria outra operação importante a realizar neste contexto de avaliação de desempenho de SFA, fundamentalmente a funcionalidade "write-back" da componente *cache*. No entanto, tal não é viável usando o mesmo método de transferência de dados.

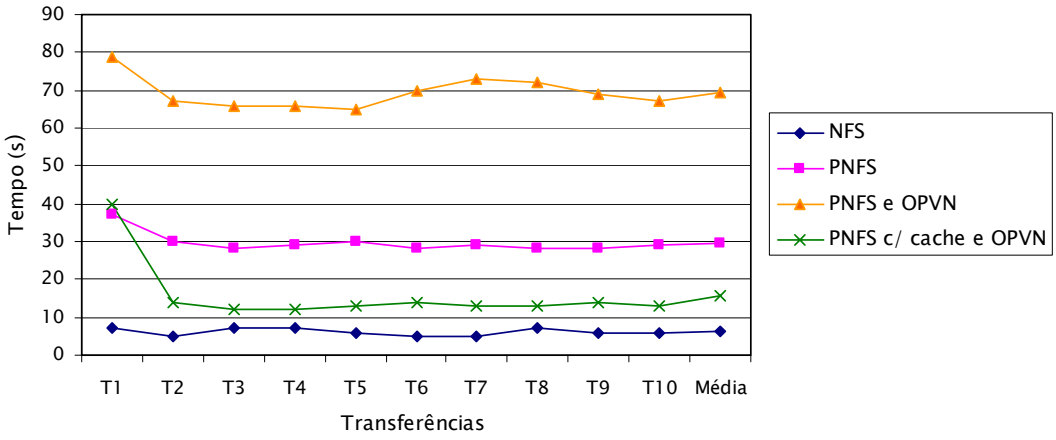

Figura 6.12 – Gráfico com as transferências nos 4 cenários.

As ferramentas de análise de desempenho de sistemas de ficheiros já citadas – IOzone e Bonnie – entre outras, implementam diversas formas de leitura e escrita (em bloco, carácter, sequencial, etc.). Permitiriam uma avaliação completa, mas como se referiu anteriormente não se mostraram adequadas a máquinas virtuais, como é o caso do *multi-cluster* de desenvolvimento.

Contudo esta avaliação parcial do desempenho de SFA é relevante para a compreensão do seu funcionamento global e de cada um dos seus componentes.

6.4 Serviço de *Cluster* Virtual

Acabou-se de implementar num ambiente de testes uma infra-estrutura computacional composta por três *clusters*, cujos os nós comunicam directamente entre si e que podem partilhar também entre si partes de sistemas ficheiros, dando a cada utilizador comum a mais que um *cluster* a noção de um sistema de ficheiros uno, transversal aos *clusters* que utiliza.

No entanto, o conceito "*Cluster* Virtual" (CV) introduzido no Capítulo 5 propõe uma utilização do *multi-cluster* próxima da *Grid*. Isto é, propõe o fornecimento de um serviço a todos aqueles que pontualmente, ou não, precisam de correr aplicações sobre recursos consideráveis ou peculiares.

Assim, a cedência de um *cluster* virtual a um utilizador pressupõe a elaboração de um conjunto de tarefas, desde que o utilizador acede ao serviço até ao momento em que é disponibilizado o CV especificado pelo utilizador. Nas próximas secções iremos detalhar essas tarefas que conduzem à construção de um *cluster* virtual para o utilizador José Silva, supondo agora que não tem acesso a nenhum *cluster* mas pretende usar os recursos da plataforma *multi-cluster*.

Toda a informação relativa ao Serviço *Cluster* Virtual é armazenada na base de dados MySQL "Multicluster".

6.4.1 Registo de utilizadores

O utilizador típico do CV não terá contas de acesso próprias no *multi-cluster* mas pretende ter, por um determinado período de tempo, recursos que aparentando um *cluster* normal lhe permitam correr aplicações já desenvolvidas, sem fazer a alterações a estas.

Assim, a primeira acção é da responsabilidade do utilizador, consistindo no seu registo no portal de suporte ao Serviço *Cluster* Virtual (SCV), ficticiamente no endereço `https://www.clustervirtual.org/registo`.

Através de um formulário o utilizador introduz dados referentes à sua identificação: Nome, Instituição, Área de Investigação. No mesmo formulário Web é-lhe fornecido um identificador, neste caso UCV1 (Utilizador de *Cluster* Virtual 1), que complementado com uma senha, a indicar pelo utilizador, constituem as suas credenciais no contexto do SCV. Ainda usando o mesmo formulário o utilizador envia a sua chave RSA pública que fica armazenada como `ucv1_rsa.pub`, e que permitirá a UCV1 aceder ao nó frontal dos seus CVs.

Tabela 6.4 – Tabela "utilizadores" do SCV.

Campo	Tipo
UCV	int(4)
Nome	varchar(40)
Email	varchar(50)
Area	varchar(20)
Instituição	varchar(40)
Senha	varchar(10)
Chave_pub	varchar(15)
Chave_int_priv	varchar(15)
Chave_int_pub	varchar(15)

Também nesta fase de registo é gerado um par de chaves RSA para o utilizador UCV1 que só serão válidas no âmbito do SCV. Este par de chaves identificará o utilizador UCV1 no acesso aos vários nós de computação dos seus futuros CVs.

Toda a informação gerada no registo dos utilizadores é armazenada na tabela "utilizadores" que integra a base de dados "Multicluster", descrita na Tabela 6.4, enquanto um exemplo do registo de um utilizador é apresentado na Tabela 6.5.

O registo só é efectuado uma vez. Os futuros acessos ao SCV via portal são autenticados por ID e senha do utilizador.

Tabela 6.5 – Exemplo do registo de um utilizador.

Campo	Valor
UCV	10
Nome	José Silva
Email	jsilva@instituicao.pt
Area	Física dos cristais
Instituição	Instituto da Física Aplicada
Senha	########
Chave_pub	ucv10_rsa.pub
Chave_int_priv	ucv10_rsa_int.pri
Chave_int_pub	ucv10_rsa_int.pub

6.4.2 Atribuição de recursos

Sempre que um utilizador pretender utilizar o SCV primeiro tem de especificar os recursos que pretende para o seu CV. Para isso, depois de autenticado no portal, é-lhe disponibilizada uma lista de recursos do *multi-cluster* disponíveis no momento.

Continuando a exemplificação com UCV10, quando este acede ao portal e solicita recursos ao SCV, é-lhe apresentada a Tabela 6.6 no portal.

Tabela 6.6 – Recursos disponíveis no *multi-cluster*.

Cluster	C1	C1	C2	C2	C3	C3
Nó	N01	N21	N02	N22	N03	N23
Processador	Intel Xeon	Intel Xeon	ADM Turion	ADM Turion	Intel Xeon	Intel Xeon
Memória	2 GB	2 GB	4 GB	4 GB	2 GB	2 GB
Rede Interna	Fast Ethernet	Fast Ethernet	Gigabit Ethernet	Gigabit Ethernet	Fast Ethernet	Fast Ethernet
Rede Externa	Fast Ethernet	-	Ethernet	-	Fast Ethernet	-

Estes recursos livres podem ser obtidos com ajuda do escalonador de cada *cluster*. No caso do PBS a informação pode ser obtida com o comando `pbsnodes`.

Executando a sequência de comandos em N01:

```
$ pbsnodes -a | egrep -A 1 -B 4
```

Obtém-se os nós livres, sem execução de trabalhos. Depois esta informação é complementada com pesquisas às bases de dados "Cluster" de cada *cluster*.

Perante estes dados o utilizador José Silva seleccionou os nós N01, N21 e N23, com base no critério de nós com características semelhantes e com o melhor

desempenho de rede possível, para o qual se tem em consideração a capacidade rede externa uma vez que os nós de C1 e C3 (*clusters* envolvidos) serão interligados através desta rede externa.

Uma vez aceite a especificação feita pelo utilizador, é necessário reservar os nós seleccionados em cada *cluster* para que não sejam usados para execução de trabalhos no âmbito da actividade dos respectivos *clusters*. No caso do escalonador PBS, do actual exemplo, tal é realizado executando,

Em N01:

```
#pbsnodes –o N21;
```

Em N03:

```
#pbsnodes –o N23.
```

Este comando retira os nós do escalonamento para execução de trabalhos. Terminado o CV, os nós são devolvidos através do comando:

```
#pbsnodes –c nó
```

6.4.3 Utilizadores Lógicos e portos UDP

Da especificação feita pelo utilizador determina-se quais os *clusters* que são envolvidos na construção do respectivo CV: C1 e C3. Para cada um destes *clusters* é atribuído um Utilizador Lógico, consultando a tabela "utilizadores_logicos" da base de dados "Multicluster", cuja definição está representada na Tabela 6.7.

Tabela 6.7 – Tabela de Utilizadores Lógicos

Campo	Tipo
ID	int(3)
cluster	varchar(15)
conta	varchar(10)
uid	int(3)
gid	int(3)
estado	int(1)

Esta tabela no *multi-cluster* de desenvolvimento está preenchida com cinco contas UL por *cluster*. Os UIDs e GIDs são iguais, diferindo de *cluster* para *cluster* do seguinte modo: C1 – de 611 a 615; C2 – 621 a 625; C3 – 631 a 635.

Fazendo uma consulta SQL à tabela com objectivo de obter um UL livre de C1 obtém-se: `ul11` (uid=611,gid=611). A mesma consulta para C3 determina: `ul13` (uid=631,gid=631).

Cada uma destas contas terá associado um *pnfsn2*, constituído por dois processos `pvfs.mountd` e `pvfs.nfsd` que comunicam, por um lado, com os processos homólogos de *pnfsn1*, por outro lado, com os clientes NFS, `mount`. Nestas comunicações são utilizados portos UDP, de preferência não atribuídos pelo IANA

(*Internet Assigned Numbers Authority*) [49]. A consulta de portos livres no IANA, determinou que:

- Portos UDP para *pnfsn1*: `pvfs.mountd` – 34001 e `pvfs.nfsd` – 34002;
- Portos UDP para *pnfsn2*: `pvfs.mountd` – gama {39700 a 40699} e `pvfs.nfsd` – gama {46000 a 46999}. Para definir os portos de cada *pnfsn2* dos ULs, institui-se a seguinte regra: somar ao primeiro porto de cada gama o UID de UL.

Aplicando esta regra aos ULs, `ul11` e `ul13`, conclui-se que:

- Portos para `pnfsn2.ul11` (C1): `pvfs.mountd` – 40311 e `pvfs.nfsd` – 46611.
- Portos para `pnfsn2.ul13` (C3): `pvfs.mountd` – 40331 e `pvfs.nfsd` – 46613.

Recorda-se que cada um destes dois *pnfsn2* corre num nó frontal diferente, pelo que os seus portos UDP poderiam coincidir.

6.4.4 Representação lógica

Nesta altura tem-se praticamente toda a informação para definir a Representação Lógica (RL) de CV10 que é constituído pelo nó frontal N01 e pelos nós de computação N21 e N23. Falta apenas obter os endereços IP – privado e público – do nó frontal, os endereços IP dos nós de computação e respectivas redes privadas. Esta informação é obtida através de consultas SQL às tabelas "nodes" e "networks" das bases de dados "Cluster" de C1 e C3.

Assim, sobre C1 obtém-se:

- Rede privada – 10.1.1.0;
- Mascara – 255.255.255.0.

Sobre o nó frontal o resultado da consulta é:

- IP público – 192.168.10.1;
- IP privado – 10.1.1.1.

Já sobre os nós de computação a informação obtida consiste em:

- IP de N21 – 10.1.1.253;
- IP de N23 – 10.3.3.253.

Com toda esta informação é agora possível construir a RL de CV10. É feita na linguagem de notação XML [50] e é guardada no ficheiro `cv10.rl`. Está representada na Figura 6.13.

```
<!—-Representacao logica de CV10 -->
<!—- Informacao do no frontal de CV10 -->
    <no_frontral>
        <nome>n01</nome>
        <ip_pub>192.168.10.1</ip_pub>
        <ip_pri>10.1.1.1</ip_pri>
        <cluster> C1
            <rede_ip_priv>10.1.1.0</rede_ip_priv>
            <rede_ip_priv_mask>255.255.255.0</rede_ip_priv_mask>
        </cluster>
        <sfa>/export/home/sfa/ul11
            <uid>600</uid>
            <gid>600</gid>
        </sfa>
    </no_frontral>
<!—- Informacao de C1 participante em CV10 -->
    <cluster> C1
        <rede_ip_priv>10.1.1.0</rede_ip_priv>
        <rede_ip_priv_mask>255.255.255.0</rede_ip_priv_mask>
        <no>N21
            <ip>10.1.1.253</ip>
        </no>
        <ul>ul11
            <uid>611</uid>
            <gid>611</gid>
        </ul>
        <pnfsn2>
            <porto_mountd>51611</porto_mountd>
            <porto_nfsd>52612</porto_nfsd>
        </pnfsn2>
    </cluster>
<!—- Informacao de C3 participante em CV10 -->
    <cluster> C3
        <rede_ip_priv>10.3.3.0</rede_ip_priv>
        <rede_ip_priv_mask>255.255.255.0</rede_ip_priv_mask>
        <no>N23
            <ip>10.3.3.253</ip>
        </no>
        <ul>ul13
            <uid>631</uid>
            <gid>631</gid>
        </ul>
        <pnfsn2>
            <porto_mountd>51631</porto_mountd>
            <porto_nfsd>51631</porto_nfsd>
        </pnfsn2>
    </cluster>
```

Figura 6.13 – Representação Lógica de CV10.

6.4.5 Implementação do sistema de ficheiros

A partir daqui SCV avança para a construção do sistema de ficheiros de CV10, realizando duas tarefas: 1) activar os procuradores de nível 2, `pnfsn2.ul11` e `pnfsn2.ul13`, respectivamente em N01 e N03; 2) activar clientes NFS em N21 e N23. A estas duas tarefas estão subjacentes as respectivas configurações para as quais RL fornece toda informação.

Começando pela tarefa 1, SCV constrói os ficheiros de configuração para os *pnfsn2*, conforme se apresentam de seguida.

Configurações de `pnfsn2.ul11`:

- n2_exports_mountd.ul11

```
/export/home/sfa/ul11 localhost 10.1.1.0/255.255.255.0
```

- n2_exports_nfsd.ul11

```
/export/home/sfa/ul11 \
localhost(rw,map_static=/etc/pvfs/n2_map.ul11) \
10.1.1.0/255.255.255.0(rw,map_static=/etc/pvfs/n2_map.ul11)
```

- n2_map.ul11

```
uid 611 600
gid 611 600
```

Configurações de `pnfsn2.ul13`:

- n2_exports_mountd.ul13

```
/export/home/sfa/ul11 localhost 10.3.3.0/255.255.255.0
```

- n2_exports_nfsd.ul11

```
/export/home/sfa/ul13 \
localhost(rw,map_static=/etc/pvfs/n2_map.ul13) \
10.3.3.0/255.255.255.0(rw,map_static=/etc/pvfs/n2_map.ul13)
```

- n2_map.ul13

```
uid 631 600
gid 631 600
```

O par 600/600 refere-se ao UID/GID de `/export/home/sfa` já definido no SFA, no início destes capítulo.

Elaboradas as configurações, SCV levanta os procuradores de níveis 2 nos nós frontais, N01 e N03, usando a *script* `pnfsn2.pl` com os seguintes parâmetros:

– U: utilizador lógico;

– N: porto UDP de `pvfs.nfsd` de *pnfsn2*;

– M: porto UDP de `pvfs.mountd` de *pnfsn2*;

– S: IP privado do servidor que corre *pnfsn1*.

Assim *pnfsn2* é activado em N01 e N03, respectivamente por:

- `pnfsn2.pl –U ul11 –N 52611 –M 51611 –S 127.0.0.1`;
- `pnfsn2.pl –U ul13 –N 52631 –M 51631 –S 10.1.1.1`;

Concluída a tarefa 1, SVC avança para a tarefa 2, ou seja, para a activação dos clientes NFS em N01, nó frontal, e em N21 e N23, nós de computação de CV10. Para tal utiliza-se a *script* `cnfs.pl` (Cliente NFS) com os seguintes parâmetros de entrada:

- O: utilizador lógico na origem, aquele que partilha `/export/home/sfa/ul`$_{ic}$;

- D: utilizador lógico no destino, aquele que monta a directoria referida em cima;

- S: IP privado do servidor, nó frontal, que corre *pnfsn2*;

- N: porto UDP de `pvfs.nfsd` de *pnfsn2*;

- M: porto UDP de `pvfs.mountd` de *pnfsn2*;

Assim, a execução de `cnfs.pl` em N01 toma a forma:

- `cnfs.pl -O ul11 -D ul11 -N 52611 -M 51611 -S 127.0.0.1;`

Em N21 a execução é idêntica, apenas muda o servidor:

- `cnfs.pl -O ul11 -D ul11 -N 52611 -M 51611 -S 10.1.1.1;`

Enquanto em N23 a execução corresponde a:

- `cnfs.pl -O ul11 -D ul13 -N 52631 -M 51631 -S 10.3.3.1.`

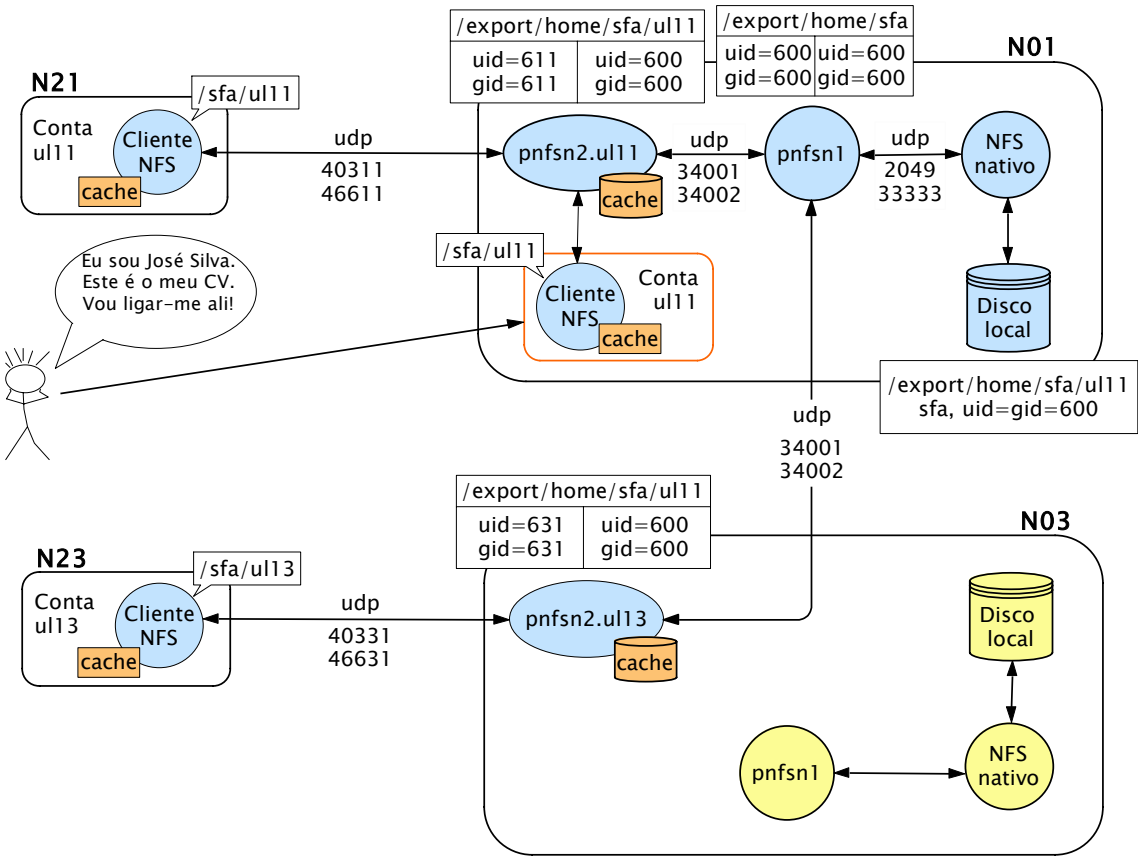

Figura 6.14 – Sistema de ficheiros de CV10.

A utilização de `cnfs.pl` corresponde à utilização do comando `mount` com a respectiva parametrização, que em cada um destes casos é a seguinte:

- ```
  mount -t nfs -o port=52611,mountport=51611,vers=2, \
  mountvers=2,udp,rsize=8192,wsize=8192 \
  127.0.0.1:/export/home/sfa/ul11 /sfa/ul11
  ```
  ```

- `mount -t nfs -o port=52611,mountport=51611,vers=2, \ mountvers=2,udp, rsize=8192,wsize=8192 \ 10.1.1.1:/export/home/sfa/ul11 /sfa/ul11`
- `mount -t nfs -o port=52631,mountport=51631,vers=2, \ mountvers=2,udp, rsize=8192,wsize=8192 \ 10.3.3.1:/export/home/sfa/ul11 /sfa/ul13`

Nota-se que toda a informação necessária para a parametrização de *scripts* é extraída da RL do *cluster* virtual.

Neste momento está implementado o sistema de ficheiros de CV10, ilustrado na

Figura 6.14, que se baseia na directoria `/export/home/sfa/ul11` sedeada em N01 partilhada por NFS para o próprio N01 e para os nós de computação N21 e N23.

6.4.6 Associação de identidade

Tem-se portanto nesta fase um *cluster*, mas ainda não está disponível para o utilizador José Silva que o solicitou. Pode dizer-se que a sua identidade é anónima. Foi construído com utilizadores lógicos e ainda não tem um proprietário real. É assim necessário associar a CV10 a identidade de José Silva.

No essencial esta associação consiste em três operações: 1) Comutação da directoria de trabalho de cada UL de CV10: `ul11` e `ul13`, 2) Instalação das chaves RSA internas – privada e pública – de UCV10, 3) Instalação da chave RSA publica de UCV10.

1) Comutação da directoria de trabalho cada UL de CV10

Cada um dos ULs tem a sua directoria de trabalho definida em `/export/home` do seu *cluster* Rocks. No entanto, como já se viu anteriormente, com a criação do sistema de ficheiros de CV passaram a existir novas directorias de trabalho *paralelas* para cada UL, definidas em `/sfa` mas apenas nos nós do CV, assim:

- N01 – `/sfa/ul11`
- N21 – `/sfa/ul11`
- N23 – `/sfa/ul13`

Pretende-se que estas passem a ser as directorias de trabalho de `ul11` e `ul13` no contexto de CV10. Para concretizar esta mudança é utilizado o comando `usermod` conjugado com o serviço 411 do Rocks.

Em N01 de C1 executa-se:

```
# usermod –d /sfa/ul11 ul11
# make –C /var/411
```

O primeiro comando altera a directoria de trabalho de `ul11` definida em `/etc/passwd`, e o segundo força o 411 a actualizar esta alteração em todos os nós de C1. No entanto, esta nova directoria de trabalho só existe N01 e N21. Em N11 `ul11` não tem agora directoria de trabalho.

Repetindo estes comandos em N03 de C3, tem-se a comutação de directorias concluída.

2) Instalação das chaves RSA internas de UCV10

As chaves privadas internas foram criadas na fase de registo de UCV10 e encontram-se armazenadas em `/opt/mc/chaves` de N01.

Estando as directorias de trabalho, em todos os nós de CV10, sintonizadas na directoria de UCV10, definida em `/export/home/sfa/ul11` de N01, a instalação das chaves RSA privadas corresponde a replicá-las para a sub-directoria `.ssh`. Depois de consultar a tabela "utilizadores" da base de dados "Multicluster" determina-se que a instalação das chaves implica a seguinte réplica de ficheiros:

```
N01:/opt/mc/chaves/ucv10_rsa_int.pri
                ↓
N01:/export/home/sfa/ul11/.ssh/id_rsa
```

```
N01:/opt/mc/chaves/ucv10_rsa_int.pub
                ↓
N01:/export/home/sfa/ul11/.ssh/id_rsa.pub
```

Esta réplica é realizada dentro do mesmo nó, bastando uma simples cópia. Caso fosse entre nós deferentes, a réplica seria feita de forma segura com o comando `scp`. Depois da réplica é necessário actualizar o ficheiro `authorized_keys` com a chave RSA interna pública. Após isto é possível estabelecer sessões SSH entre os nós do CV10, N01, N21 e N23, usando as contas de acesso `ul11` e `ul13`, tendo como método de autenticação o mecanismo de chaves públicas.

3) Instalação da chave RSA pública de UCV10

Agora é necessário garantir que o utilizador José Silva, registado como UCV10 no serviço SCV, tenha acesso interactivo, directo ao seu *Cluster* Virtual, isto é, que possa estabelecer sessões SSH com o nó frontal usando a conta `ul11`.

Para que isto aconteça é então preciso instalar na directoria de UCV10, `/export/home/sfa/ul11`, a chave pública de José Silva que se encontra armazenada no nó frontal N01 com a designação `ucv10_rsa.pub`, conforme consta na tabela "utilizadores". À semelhança do que foi feito anteriormente, esta instalação implica a seguinte replicação:

```
N01:/opt/mc/chaves/ucv10_rsa.pub
                ↓
N01:/export/home/sfa/ul11/.ssh/ucv10_rsa.pub
```

Finalmente, actualizar mais uma vez o ficheiro `authorized_keys` com esta chave. Conclui-se desta forma a associação da identidade de José Silva ao CV10, garantindo-se que só ele pode utilizar este *Cluster* Virtual.

É também importante notar que em face do sistema de ficheiros de CV10 só integrar os nós de computação atribuídos, e uma vez que se efectuou a comutação das directorias de trabalho originais dos utilizadores lógicos (ULs) para as directorias paralelas, estão assegurados os mecanismos de segurança que impedem o utilizador José Silva aceder a outros nós que constituem o *multi-cluster*.

6.4.7 Sessão interactiva

Implementada a autenticação no CV10, é agora conveniente promover condições para um acesso confortável, flexível e também imediato ao momento da especificação do CV. Sabendo-se que o acesso é interactivo, suportado por SSH, é disponibilizado ao utilizador um cliente SSH, já devidamente configurado com os seguintes parâmetros:

- Endereço IP público do nó frontal N01;
- Conta de acesso;
- Método de autenticação.

Neste âmbito, o SCV disponibiliza um cliente SSH desenvolvido em Java sobre a forma de *Applet*, designado por MindTerm [51], parametrizável e que, sendo embebido numa página HTML, pode ser descarregado automaticamente do portal do SCV e executado localmente sobre uma plataforma JRE (*Java Run Enviorement*) instalada no computador pessoal do utilizador.

Na sequência de construção de CV10, surge então a fase de elaboração de uma página HTML dinâmica do portal que incluirá o MindTerm [52] personalizado para este CV. Depois de descarregado e executado, o cliente estabelece uma sessão SSH ao nó frontal de CV10, bastando para isso que o utilizador José Silva indique qual a sua chave RSA privada.

A página com o título "Sessão SSH para CV10" desenvolvida dinamicamente em PHP inclui código dividido em três secções:

- Informação pessoal do utilizador, neste caso de José Silva;
- Informação sobre o *Cluster* Virtual CV10: Nós e respectivas características, UL usado no acesso;
- Especificação do *applet* MindTerm com a parametrização concreta para esta sessão, através do seguinte código:

```
<APPLET CODE="com.mindbright.application.MindTerm.class"
<ARCHIVE="mindterm.jar" WIDTH=0 HEIGHT=0\>
<PARAM NAME="cabinets" VALUE="mindterm.cab">
<PARAM NAME="username" value="ul11">
<PARAM NAME="authmetod" value="keypublic">
<PARAM NAME="server" value="192.168.10.1">
<PARAM NAME="sepframe" value="true">
<PARAM NAME="debug" value="true">
<PARAM NAME="allow-new-server" value="false">
</APPLET>
```

Essencialmente definem-se os parâmetros:

- Conta de acesso, `username=ul11`;
- Método de autenticação, `authmetod=keypublic`;
- Endereço público do servidor, `server=192.168.1.1`;
- Execução do *applet* numa nova janela, `sepframe=true`.

Assim que esta página é acedida pelo programa de navegação Web (Internet Explorer, Firefox, etc.), o *applet* é transferido e executado no ambiente JRE do computador pessoal do utilizador. A execução é realizada com os parâmetros acima definidos, de modo que o utilizador apenas necessita de indicar a localização da sua chave RSA privada no seu computador. Concluída esta operação, é estabelecida a sessão SSH.

6.4.8 Ambientes para aplicações paralelas

MindTerm dispõe de módulos SCP e SFTP que permitirão transferir informação do computador pessoal do utilizador para o nó frontal do CV10. Da mesma forma, o utilizador pode transferir dados, aplicações ou mesmo ambientes de programação, como é o caso do PVM. Na Figura 6.15 mostra-se a transferência de um conjunto de dados num formato *zip*.

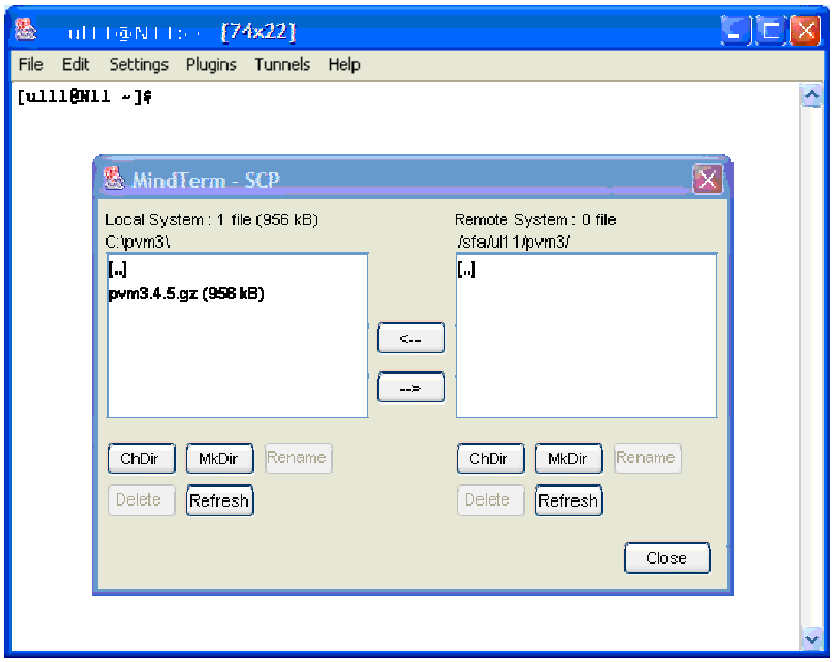

Figura 6.15 – Transferência de ficheiros com SCP do MindTerm.

Preparação do ambiente PVM

Como exemplo da utilização do CV10, executam-se algumas aplicações que acompanham a distribuição do PVM 3.0 para Linux. Assim, descarrega-se esta versão para a directoria `pvm3`, localizada na directoria de trabalho de UCV10. De seguida ajustam-se algumas variáveis de ambiente do sistema operativo como `PVM_ROOT` e `PVM_ARCH`, editando os ficheiros de configuração `.bashrc` e `.bash_profile`.

A definição do ambiente PVM passa ainda pela especificação dos nós que integram a "máquina virtual paralela" e que neste caso correspondem aos nós de CV10. Assim o ficheiro `$HOME/nos_pvm.cfg`, em N01, contém essa informação:

```
10.1.1.1     lo=ul11
10.1.1.253   lo=ul11
10.3.3.253   lo=ul13
```

Usando-se SSH (em vez do RSH) na activação remota dos processos `pvmd`, é ainda necessário complementar o ambiente PVM com a indicação dos ULs dos respectivos nós de computação de CV10. Isto é realizado colocando a variável `lo`, como se indica em cima, ou então, editando o ficheiro `$HOME/.ssh/config` de `ul11` em N01, e introduzindo a seguinte informação:

```
Host 10.1.1.253
User ul11
Host 10.3.3.253
User ul13
```

Com estes passos está definido o ambiente PVM, pronto a executar aplicações paralelas desenvolvidas para este ambiente.

Execução de aplicações Paralelas em ambiente PVM

A execução de aplicações em ambiente PVM inicia-se pela activação dos processos que o suportam, ou seja, pela simples activação da consola em N01, automaticamente se constitui a "máquina paralela" com os nós definidos em `$HOME/nos_pvm.cfg`.

Alternativamente, esta "máquina" pode ser construída a partir da consola:

```
$pvm> add 10.1.1.1
$pvm> add 10.1.1.253
$pvm> add 10.3.3.253
```

Concluído todo este processo de configuração do ambiente PVM é possível executar aplicações em CV10, exactamente como o utilizador o faz num *cluster* físico.

Na Figura 6.16 apresenta-se uma sessão em que se executa no CV10 a aplicação "hello world", em que cada nó vai emitindo a mensagem "*hello, world from…*".

A utilização do MPI no contexto do CV é muito semelhante à que acaba de ser descrita para o PVM, pelo que não é relevante a sua abordagem no momento.

```
$pvm> conf
3 hosts, 1 data format
            HOST    DTID    ARCH    SPEED        DSIG
             N01   40000   LINUX     1000  0x00408841
             N11   80000   LINUX     1000  0x00408841
             N13   c0000   LINUX     1000  0x00408841
$pvm>
$pvm> spawn -> hello
[10:t4002b] i'm t4002b
[10:t4002b] from t4002c: hello, world from N01
[10:t4002b] EOF
[10] finished
$pvm>
$pvm> spawn -> hello
[11:t4002d] i'm t4002d
[11:t4002d] from t80003: hello, world from N11
[11:t4002d] EOF
[11] finished
$pvm>
$pvm> spawn -> hello
[12:t4002e] i'm t4002e
[12:t4002e] from tc0003: hello, world from N13
[12:t4002e] EOF
[12] finished
```

Figura 6.16- Execução de aplicação em ambiente PVM.

6.5 Experiência

A terminar este capítulo relata-se uma experiência, entre outras, realizada num cenário muito próximo da realidade, esquematizado na Figura 6.17. A experiência teve como objectivo validar as várias componentes da solução global que é proposta no presente trabalho, tendo por base infra-estruturas de comunicaões e de computação reais.

Como se mostra na Figura 6.17, utilizaram-se dois *clusters*: C1 formado por três nós instalados em máquinas virtuais, concretamente, é um dos *clusters* usados na plataforma de desenvolvimento; o outro, C2, constituído por 8 nós iguais, cada um equipado com bi-processador Intel Xeon a 3 GHz, 2 GB de memória, dois discos SATA de 80 e 160 GB, e uma interface de rede *Gigabit Ethernet* (o nó frontal possui duas interfaces de rede).

A primeira tarefa consistiu na interligação física dos dois *clusters*. Cada *cluster* tem a sua rede privada que liga os nós de computação e uma ligação ao exterior através do respectivo nó frontal. C1 encontra-se inserido numa rede residencial, privada (192.168.123.0/24), ligada por sua vez a um Operador de comunicações através de encaminhador IP com o endereço público 83.132.67.187. O encaminhador assegura também o serviço NAT, mapeando toda a comunicação com origem na rede residencial, incluindo a rede privada de C1 10.1.1.0/24), naquele endereço IP público.

O *cluster* C2 dispõe de uma rede privada IP (10.10.10.0/24) sobre *Gigabit Ethernet* a qual se interliga à rede da Universidade do Minho através do seu nó frontal que tem o endereço IP público 193.136.19.163.

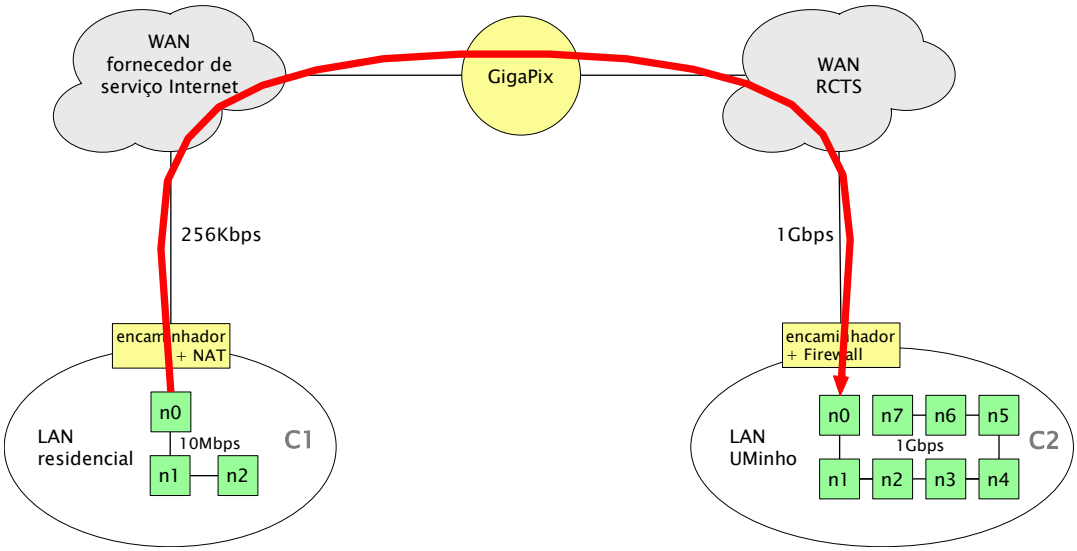

Figura 6.17 – Cenário de uma experiência.

Os dois *clusters* constituem apenas um Domínio OpenVPN (DOVPN-1) em que C2 é o servidor e C1 é cliente. A interligação consistiu na atribuição de certificados digitais aos dois *clusters* e nas respectivas configurações OpenVPN conforme foi especificado no Capítulo 4. Assim, a interligação de C1 e C2 concretizou-se pela activação dos processos OpenVPN, em ambos os nós frontais, que estabeleceram um túnel UDP/IP. De realçar que este túnel atravessou um encaminhador com NAT e várias infra-estruturas de comunicações, algumas delas com *firewalls* instalados: rede do operador, GigaPIX (ponto de interligação de redes portuguesas), RCTS (rede académica portuguesa) e por último, a rede da Universidade do Minho.

Concluída a interligação dos *clusters* a tarefa seguinte constou da instalação e configuração dos procuradores NFS em ambos os nós frontais que integram o SFA da plataforma multi-*cluster*.

Com estas duas componentes em funcionamento estavam criadas as condições para a definição e implementação de CVs. Foi então definido um CV com a seguinte constituição:

CV_{teste} = {c1-n0, c1-n1,c2-n5, c2-n8}

Isto é, um *cluster* em que o nó frontal é N0 de C1 e os nós de computação são os nós N5 e N8 de C2 e ainda o nó N1 de C1. Este CV_{teste} tinha assim o seu sistema de ficheiros centralizado em C1. Como Utilizadores Lógicos foram usados: `ul11` em C1 e

`ul12` em C2. Depois de associadas as chaves RSA de um hipotético utilizador foi possível aceder ao CV através do cliente SSH em Java, MindTerm, a partir de um computador ligado na rede residencial desta experiência.

Finalmente, procedeu-se ao ajuste do ambiente PVM e foram executadas várias aplicações, exactamente como se tratasse de um *cluster* real.

6.6 Conclusão

Ao longo este capítulo foram abordados os pontos mais importantes para a implementação das propostas definidas nos Capítulos 4 e 5, isto é, Plataforma *Multi-cluster* e Serviço de *Cluster* Virtual.

Partindo de uma plataforma de desenvolvimento montada com base em máquinas virtuais, foi feito um percurso exemplificativo de como atingir os diversos objectivos:

- Implementação da rede de interligação de *clusters* com base em OpenVPN;
- Concretização de um sistema de ficheiros alargado a todos os *clusters*, com minimização do impacto OpenVPN introduzindo uma hierarquia de *cache* do lado dos clientes do sistema de ficheiros;
- Implementação de serviço adicional à plataforma *multi-cluster*, designado por Serviço de *Cluster* Virtual, que permite a criação de *Clusters* Virtuais de forma dinâmica a pedido do utilizador, com aproximação ao ambiente de Grade Computacional;
- Execução de aplicações paralelas em ambiente de *Cluster* Virtual.

Identificam-se dois níveis de implementação. Um nível que corresponde à administração de elementos estáticos, como é caso da rede de interligação de *clusters* ou dos procuradores de nível 1, que pode ser entregue a uma administração transversal à plataforma *multi-cluster*, uma administração trans-domínio que opera as infra-estruturas relacionadas com o *multi-cluster*. Outro nível de administração, focado na criação de uma abstracção entre utilizador e plataforma *multi-cluster*, opera os componentes que integram o Serviço de *Cluster* Virtual, como sejam: o portal, os procuradores NFS dinâmicos de nível 2, os módulos de software que geram configurações dinâmicas e activam os CVs, entre outros.

Assim, este capítulo pretendeu dar uma visão de como se concretiza os modelos especificados anteriormente e contextualizar a integração das tecnologias identificadas no Capítulo 3.

Capítulo 7

Discussão e trabalho futuro

Desde logo se identificaram desafios essenciais para a concretização deste trabalho, nomeadamente: 1) Interligar as redes privadas dos *clusters* através de redes públicas protegidas por sistemas de controlo de tráfego e anti-intrusão, 2) Recriar sobre o *multi-cluster* a imagem de sistema uno, capaz de executar aplicações baseadas na passagem de mensagens e de fornecer um sistema de ficheiros para partilha de dados, 3) autenticar no *multi-cluster* utilizadores dos diferentes domínios de administração.

Cada um daqueles desafios implicou a resolução de sucessivos problemas e alterações de estratégia, mas foi possível projectar uma solução que responde ao objectivo inicialmente definido, tendo sido demonstrados os diversos aspectos da sua implementação.

Esta solução pode agora ser vista como a sobreposição de três níveis: 1) rede virtual que interliga de forma segura todos os *clusters* participantes na plataforma *multi-cluster*, garantindo total conectividade entre todos os nós, 2) sistema de ficheiros alargado que permite a cada utilizador e às suas aplicações o acesso aos seus dados distribuídos pelos *clusters* a partir de qualquer nó, 3) serviço de *cluster* virtual que disponibiliza de forma dinâmica a definição e implementação de ambientes de execução personalizados, de acordo com as especificações realizadas pelo utilizador.

7.1 Conclusões

Este trabalho permitiu concluir da viabilidade de se constituírem comunidades de utilizadores, Organizações Virtuais na terminologia *Grid*, partilhando recursos de computação com o objectivo de os rentabilizar, sem que isso implique alterações na arquitectura das suas plataformas locais de computação.

Deve-se ainda realçar que tal se conseguiu pela integração de várias tecnologias perfeitamente estabilizadas – VPN, TLS, NFS, SSH, MySQL, Web, entre outras – implementadas através de aplicações de código aberto. Paralelamente, o trabalho também demonstrou que este compromisso para minimizar alterações nas plataformas já existentes, tem um custo que se reflecte no desempenho global da solução proposta, nomeadamente ao nível das comunicações e do sistema de ficheiros. Este factor do desempenho deve ser considerado, podendo condicionar a aplicação da solução em contextos muito exigentes.

Como principais contributos deste trabalho destacam-se: 1) a arquitectura de rede virtual IP para interligar *clusters* sobre redes públicas, fundada com tecnologia VPN com suporte TLS, projectada como uma progressão aritmética de túneis UDP agrupados em Domínios OpenVPN e garantindo total conectividade entre nós, 2) a definição

baseada em NFS da arquitectura de um Sistema de Ficheiros Alargado abrangendo *clusters* de diferentes domínios de administração, e 3) o conceito de *cluster* virtual concretizado como um ambiente de execução (re)definido dinamicamente pelo utilizador sobre uma plataforma de computação alargada heterogénea, constituindo uma imagem de sistema uno com acesso directo, interactivo via Web.

Uma conclusão secundária, mas que importa referir, diz respeito às vantagens da tecnologia de virtualização aplicado num ambiente de desenvolvimento. No trabalho usaram-se *clusters* sobre máquinas virtuais VMware, tendo-se constatado da versatilidade e flexibilidade desta tecnologia que permitiu a simulação de vários cenários com grande comodidade.

7.2 Trabalho futuro

Como já foi referido, com este trabalho foi possível projectar uma "Infra-estrutura para a Computação *Multi-cluster* em Ambiente *Grid*", partindo de vários *clusters Beowulf* independentes, composta por três níveis: Rede Virtual, Sistema de Ficheiros Alargado e Serviço de *Cluster* Virtual.

O Serviço de *Cluster* Virtual (SCV) constitui uma componente fundamental em termos de trabalho futuro. De facto este serviço pode registar grandes progressos pela introdução de várias funcionalidades que automatize e enriqueça as suas acções.

Identificam-se duas áreas do SCV que podem ser melhoradas: negociação de recursos de CV e gestão de CVs. Na actual especificação de SCV a atribuição de recursos é realizada, em certa medida, pelo utilizador mediante os recursos disponíveis no momento apresentados pelo serviço. Uma outra abordagem pode consistir em a atribuição de recursos ser efectuada completamente pelo SCV de acordo com a especificação do utilizador.

Em primeiro lugar, isto implica que o utilizador tenha conhecimento do tipo de recursos existentes na plataforma *multi-cluster* (processadores, memórias, discos, etc). Em segundo lugar, dispor de um mecanismo que utilize a especificação de CV submetida pelo utilizador para descobrir os recursos necessários na plataforma e proceda à sua reserva caso se encontrem livres. Caso tal não aconteça, deve ser proposto ao utilizador a configuração mais aproximada no momento.

Este conjunto de funções pode corresponder a uma futura componente do SVC designada por Gestor de Recursos (GR) que terá como missão: a reserva dinâmica de recursos do CV especificado pelo utilizador. Constitui um módulo de software que tem como parâmetros de entrada uma especificação de CV fornecida pelo portal de SCV e devolve como resultado a Representação Lógica (RL) do *cluster* virtual

Relativamente à gestão de CVs considera-se importante a especificação e respectiva implementação de um Gestor de *Clusters* Virtuais (GCV). Este componente terá como principais funções: 1) implementar cada CV a partir da sua RL, 2) monitorizar os CVs activos e 3) Desactivar CVs. Desta forma perspectiva-se GCV como a componente principal do Serviço de *Cluster* Virtual, funcionando como um escalonador de CVs sobre a plataforma *multi-cluster*.

Concluída a RL de um *cluster* virtual, GCV terá a responsabilidade de implementar o CV, criando o respectivo sistema de ficheiros, associando a identidade do utilizador a CV e disponibilizando a sessão interactiva em CV. Quando o utilizador indicar o fim da sua sessão de trabalho, GCV deverá desfazer o CV, libertando os respectivos recursos. Complementarmente, GCV deve fornecer à administração da plataforma *multi-cluster* informação sobre os CVs activos.

Na Figura 7.1 mostra-se um possível diagrama funcional de SCV que pode ser considerado um ponto de partida para trabalho futuro.

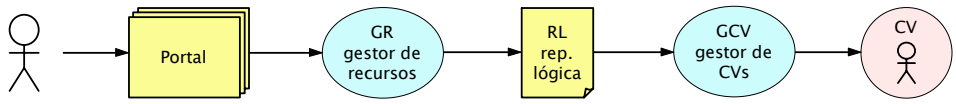

Figura 7.1 – Proposta de diagrama funcional de SCV.

A consolidação de todos os componentes desta infra-estrutura para computação *multi-cluster* é sem dúvida uma tarefa realizável. A solução passa pelo desenvolvimento de módulos de software que uma vez integrados possam ser disponibilizados sobre a forma de um pacote de software.

Referências

Capítulo 1

[1] Beowulf, http://www.beowulf.org, acesso em Março 2006.

[2] R. G. Brown, *Engineering a Beowulf-style Compute Cluster*, Duke University, May 2004

[3] M. Barreto, P. Navaux, *Estudo sobre Computação baseada em Clusters e Grids*, Porto Alegre, Brazil, 2002.

[4] W. Groop, E. Lusk and T. Sterling (Ed.), *Beowulf Cluster Computing with Linux*, Second Edition, USA, The MIT Press, 2003.

[5] J. D. Sloan, *High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI*, USA, O'Reilly, 2004.

[6] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, January 2002.

[7] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a New Computing Infrastructure*, Second Edition, Morgan Kaufmann Publishers Inc., USA, 2004.

[8] Global Grid Forum., http://www.gridforum.org, acesso em Julho 2006.

[9] Globus. http://www.globus.org, acesso em Maio 2006.

[10] Condor, http://www.cs.wisc.edu/condor/, acesso em Maio 2006.

[11] Legion, http://www.cs.virginia.edu/_legion, acesso em Maio 2006.

Capítulo 2

[12] D. Royo, N. Kapadia, J. Fortes, *Running PVM Applications in the PUNCH Wide Area Network-Computing Environment.*

[13] N.H. Kapadia, R.J.O. Figueiredo and J.A.B. Fortes, *PUNCH: Web portal for running tools*, IEEE Micro, May–June 2000.

[14] X. Zhang, K. Keahey, I. Foster, T. Freeman1, *Virtual Cluster Workspaces for Grid Applications*, University of Chicago, UEA, 2005.

[15] Figueiredo, R., P. Dinda, and J. Fortes, *A Case for Grid Computing on Virtual Machines*, 23rd International Conference on Distributed Computing Systems, 2003.

[16] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu, and X. Zhu, *From Virtualized Resources to VirtualComputing Grids: The In-VIGO System*, Future Generation Computer Systems, 2004.

[17] J. Chase, L. Grit, D. Irwin, J. Moore, and S. Sprenkle, *Dynamic Virtual Clusters in a Grid Site Manager,* 12th International Symposium on High Performance Distributed Computing (HPDC-12), 2003.

Capítulo 3

[18] VMware white paper, *Virtualization Overview*, http://www.vmware.com, acesso em 2005.

[19] VMware white paper, *Virtualization: Architectural Considerations and Other Evaluation Criteria*, http://www.vmware.com, acesso em 2005.

[20] Barham, P., B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebar, I. Pratt, and A. Warfield. *Xen and the Art of Virtualization*, ACM Symposium on Operating Systems Principles.

[21] TOP500 Supercomputer, http://www.top500.org**.**

[22] R. Ali, R. Gupta, G. Kochhar, and B. Bryce, *Platform Rocks: A Cluster Software Package for Dell HPC Platforms*, http://www.dell.com.

[23] *NPACI Rocks Cluster Distribution: Users Guide*, http://www.rocksclusters.org/.

[24] P. M. Papadopoulos, M. J. Katz, and G. Bruno. *NPACI Rocks: Tools and techniques for easily deploying manageable Linux clusters*, IEEE Cluster, October 2001.

[25] S. K. Baum, *Whats's in NPACI Rocks and how do I use it?*, Texas A&M University, Mar 2004.

[26] J. D. Sloan, *High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI*, USA, O'Reilly, 2004, pp: 132-149.

[27] F. Sacerdoti1, S. Chandra, K. Bhatia, *Grid Systems Deployment & Management using Rocks*, University of California, San Diego.

[28] C. Scott, P. Wolfe, M. Erwin, *Virtual Private Networks*, Second Edition, O'Reilly, USA, January 1999.

[29] James Yonan, *Understanding the User-Space VPN – History, Conceptual Foundations, and Practical Usage*, http://www.openvpn.net, acesso em Maio 2006.

[30] Tinc, http://www.tinc-vpn.org/, acesso Maio 2006.

[31] M. Feilner, *OpenVPN: Building and Integrating Virtual Private Networks*, Packt Publishing Ltd, 2005.

[32] S. Khanvilkar and A. Khokhar, *Virtual Private Networks: An Overview with performance evaluation,* IEEE Communication magazine, vol: 42(10), pp: 146-154, October 2004

[33] C. Hosner, *OpenVPN and the SSL VPN Revolution*, SANS Institute, 2004.

[34] H. Stern, M. Eisler, and R. Labiaga. *Managing NFS and NIS*. Second Edition. Sebastopol, CA: O'Reilly & Associates, Inc., 2001.

[35] R. J. Figueiredo, N. Kapadia, J. A. B. Fortes*, Seamless Access to Decentralized Storage Services in Computational Grids via a Virtual File System*, Cluster Computing Journal 7(2), April 2004, p113-122.

[36] M. Zhao, R. J. Figueiredo, *Proxy Managed Client-Side Disk Caching for the Virtual File System*, ACIS, University of Florida, November 2004.

[37] Brent Callaghan, *NFS Illustrated*, Addison-Wesley Longman, 2000, pp: 225-254.

Capítulo 4

[38] A.Z. Spector and M.L. Kazar, *Wide area file service and the AFS experimental system*, Unix Review 7(3) (1989).

[39] B. Callaghan, *NFS Illustrated*, Addison-Wesley Longman, 2000.

[40] *General Parallel File System* (*GPFS*), http://www-1.ibm.com/servers/eserver/clusters/software/gpfs.html.

[41] *Parallel Virtual File System (PVFS)*, http://www.pvfs.org.

[42] R. Figueiredo, *VP/GFS: An Architecture for Virtual Private Grid File Systems*, Technical Report TR-ACIS-03-001, ACIS Laboratory, Department of Electrical and Computer Engineering, University of Florida, May 2003.

Capítulo 5

[43] I. Foster, C. Kesselman, and S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, International Journal on Supercomputer Applications, 15(3), 2001.

[44] *Virtual Machine (PVM),* http://www.netlib.org/pvm3/.

[45] *Message Passing Interface (MPI),* http://www.lam-mpi.org.

Capítulo 5

[46] OpenVPN, *OpenVPN 2.0 HOWTO*, http://openvpn.net/howto.html, ultimo acesso Maio 2006.

[47] IOzone Filesystem Benchmark, http://www.iozone.org/, ultimo acesso Abril 2006.

[48] http://www.textuality.com/bonnie/, ultimo acesso Abril 2006.

[49] IANA, http://www.iana.org/assignments/port-numbers, acesso em Junho 2006.

[50] J. C. Ramalho, P. Henriques, *XML & XSL da Teoria à Prática*, FCA - Editora Informática, Lisboa, Outubro 2002.

[51] Appgate, *MindTerm*, http://www.appgate.com/mindterm, acesso Abril 2006.

[52] *MindTerm2.0User's Guide*, www-vlsi.stanford.edu/ssh/MindTerm_User.pdf, acesso Abril 2006.