Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Linux Terminal Server Project
Análise de desempenho

Diego Simões Carnivali

JUIZ DE FORA
AGOSTO, 2013

Linux Terminal Server Project
Análise de desempenho

Diego Simões Carnivali

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: Eduardo Pagani Julio

JUIZ DE FORA
AGOSTO, 2013

LINUX TERMINAL SERVER PROJECT

Análise de desempenho

Diego Simões Carnivali

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Eduardo Pagani Julio
Me

Eduardo Barrere
Dr

Tiago Machado

JUIZ DE FORA

29 DE AGOSTO, 2013

Minha mãe Rita e meu pai Jorge..

Resumo

A computação tem avançado cada vez mais rápido nos últimos anos. Na medida que avança, novas tecnologias vão sendo lançadas ao mercado. A necessidade de substituição dessas tecnologias por pessoas e empresas faz com que milhares de produtos dados como obsoletos sejam descartados todo o ano no mundo todo. Este trabalho estuda uma solução que propõe alterar este cenário e permitir que máquinas ditas como obsoletas ganhem vida útil, o *Linux Terminal Server Project* (LTSP). Esta solução incorpora diversos protocolos e tecnologias já consagradas no cenário da computação para permitir a criação de um ambiente com clientes leves. Ou seja, ela permite que estas máquinas ditas como obsoletas sejam empregadas como clientes em uma rede onde o processamento desses clientes ocorre todo no servidor, utilizando a rede para a comunicação dessas partes. Para isso, este trabalho realiza um estudo teórico sobre o LTSP e uma análise prática do desempenho dessa tecnologia.

Palavras-chave: LTSP, terminal leve, cliente leve, servidor, cliente, rede.

Abstract

Computing has advanced faster than ever in recent years. As it advances, new technologies are being released to the market. The need to replace these technologies by people and companies makes thousands of products as obsolete data are discarded every year worldwide. This paper studies a solution that proposes to change this scenario and allow machines considered obsolete gain life, the *Linux Terminal Server Project* (LTSP). It incorporates several technologies and protocols already established in computing to allow the creation of an environment with thin clients. That is, it allows these machines are said to be obsolete employed as clients in a network where the processing of all customer occurs on the server, using the network to communicate these parties. For this, this paper conducts a theoretical study on the LTSP and a practical analysis of the performance of this technology.

Keywords: LTSP, thin client, server, client, network.

Agradecimentos

Agradeço primeiramente a Deus, pois sem ele nada é possível.

Agradeço a duas pessoas que, por sonharem comigo, permitiram a realização desse sonho. Minha mãe Rita e meu pai Jorge são mais merecedores dessa conquista do que eu.

Agradeço ao meu irmão Gustavo, futuro cientista da computação, por todo o apoio e motivação que sempre me deu e pela amizade maravilhosa.

Agradeço a minha namorada Mayara que participou ao meu lado de todo o período em que estive na faculdade, pelo apoio, carinho, por ser essa namorada incrível e por entender minhas ausências.

Também agradeço a minha família, que sempre esteve ao meu lado e entendeu meus momentos de impaciência.

Agradeço a todos os meus amigos, as amizades conquistadas na UFJF e no NRC, é muito bom saber que posso contar com essas pessoas.

Agradeço a todo o corpo docente do DCC da UFJF por todos os ensinamentos, em especial ao professor Eduardo Pagani por me permitir grande ganho de experiência no NRC, pelos ensinamentos e pela paciência.

Enfim, agradeço a todos que de alguma forma contribuíram para a realização desse sonho.

Sumário

Lista de Figuras

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
NRC	Núcleo de Recursos Computacionais
ICE	Instituto de Ciências Exatas
LTSP	*Linux Terminal Server Project*
DHCP	*Dynamic Host Coniguration Protocol*
IETF	*Internet Engineering Task Force*
RFC	*Request For Comments*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
DNS	*Domain Name System*
MAC	*Media Access Control*
UDP	*User Datagram Protocol*
TFTP	*Trivial File Transfer Protocol*
PXE	*Preboot Execution Environment*
NBP	*Network Bootstrap Program*
HD	*Hard Disk*
NFS	*Network File System*
IBM	*International Business Machines*
RPC	*Remote Procedure Call*
XDMCP	*Display Manager Control Protocol*
XDM	*X Display Manager*
RDP	*Remote Desktop Protocol*
CPU	*Central Processing Unit*
GB	*Gigabyte*
MB	*Megabyte*
TCOS	*Thin Client Operating System*

1 Introdução

Na década de 70, quando os computadores eram utilizados apenas nas grandes empresas, era comum encontrarmos a existência de arquiteturas centralizadas de computação. Nestas estruturas, um servidor era o responsável pelo processamento de todos os dados da empresa. Os terminais disponíveis aos usuários, quando existiam, eram terminais burros utilizados apenas para acesso ao servidor (Tanenbaum, 2008).

Com o desenvolvimento dos computadores pessoais este cenário mudou e a arquitetura de processamento passou a ocorrer de forma mais distribuída. A utilização de servidores nas instituições passou a ser responsável apenas por prover serviços aos clientes, como atribuições de configuração de rede e distribuição do acesso à internet (Morimoto, 2005).

Porém, alguns fatores estão fazendo com que a utilização de terminais burros e de uma arquitetura mais centralizada de processamento (Nieh et al, 2000) volte a ter maior importância no cenário das organizações.

A grande modernização da tecnologia de computadores (Hannessy et al, 2003), aliada ao barateamento dessa tecnologia e a facilidade de compra da mesma, estão fazendo com que computadores sejam substituídos por pessoas e empresas não pelo motivo de funcionamento, mas por estas pessoas e empresas se verem necessitadas a sempre substituir suas tecnologias por outras mais modernas (Gordon, 2009).

Estes terminais obsoletos acabam sendo descartados e melhor seria se estes computadores descartados fossem aproveitados por pessoas que, mesmo com a maior facilidade de compra desses produtos, ainda não possuem acesso a computadores e internet (Babu et al, 2007).

Estes computadores ditos como obsoletos poderiam ser empregados como terminais burros na criação de laboratórios voltados para a comunidade, em centros de informática de colégios e universidades e até mesmo em empresas (Sorj et al, 2005).

Na Universidade Federal de Juiz de Fora (UFJF), por exemplo, foi criado um projeto de Reciclagem de Computadores no qual muitos computadores obsoletos foram

recebidos como doação. Muitos desses computadores não possuem mais desempenho suficiente para executarem os sistemas atuais, ficando difícil a utilização independente dos mesmos.

Assim, o Núcleo de Recursos Computacionais (NRC) do Instituto de Ciências Exatas (ICE) da UFJF objetiva a criação de um laboratório de informática no centro de vivência do ICE utilizando máquinas obsoletas como terminais burros, o projeto de Reciclagem de Computadores também possui interesse neste projeto pois pode ajuda-los a criar laboratórios para o aprendizado da tecnologia estudada neste trabalho.

Porém, na atualidade os computadores são designados a realizarem tarefas bem mais complexas do que nos primórdios da arquitetura de processamento centralizado. Assim, se faz necessária a realização de testes que objetivam analisar os requisitos necessários aos ambientes de processamento distribuído e terminais burros para que estes possam fornecer desempenho suficiente aos usuários. Também é necessária a análise das tecnologias que permitem a criação desses ambientes (Santos et al, 2012).

Dessa forma, este trabalho tem o objetivo de analisar o projeto *Linux Terminal Server Project* (LTSP). Este projeto permite a criação de um ambiente de *thin-clients*, ou terminais leves, denominação criada para designar aqueles terminais que não realizam o processamento de suas tarefas e dependem de um servidor para processa-las.

Para isso, os objetivos específicos deste trabalho são realizar uma análise teórica sobre este projeto e realizar uma análise prática do mesmo, analisar o seu funcionamento e o seu desempenho. Estas análises tem a intenção de conseguir obter resultados que permitam orientar pessoas que desejam criar um ambiente LTSP, quanto ao consumo dos recursos de servidor e rede desse ambiente para uma determinada quantidade de clientes.

Este trabalho se inicia com o capítulo que discute sobre os principais protocolos e as principais tecnologias utilizadas pelo LTSP. Em seguida, é realizada a descrição do funcionamento do LTSP, suas características e uma análise teórica da mesma. Neste mesmo capítulo, também é explicado o conceito de *LTSP-Cluster* e apresentada a tecnologia Epoptes.

Continuando o trabalho, serão descritos o ambiente de realização dos testes e os cenários criados para a realização dos mesmos. Todo o processo de realização de

cada cenário será explicado. Os resultados obtidos serão expostos através de gráficos e analisados. Este trabalho faz sua conclusão realizando uma análise de tudo que foi realizado, e o quanto este contribuiu para o que propunha.

2 Pressupostos teóricos

Neste capítulo são abordadas as principais tecnologias e protocolos utilizados pelo LTSP para o funcionamento de sua estrutura e para a comunicação entre clientes e servidores. Os elementos aqui apresentados são o DHCP, o TFTP, o PXE, o PXELINUX, o NFS e o XDMCP.

2.1 DHCP

O *Dynamic Host Configuration Protocol* (DHCP) é regulamentado pela *Internet Engineering Task Force* (IETF) e especificado por meio da documentação *Request For Comments* (RFC) (Bugallo et al, 1999), sendo que se encontra na RFC 2131.

Este protocolo utiliza uma arquitetura cliente-servidor na qual o servidor utiliza o DHCP para fornecer parâmetros de configuração de rede TCP/IP aos seus clientes, como o endereço IP do cliente, o endereço IP do roteador em que o cliente está conectado, informações e configurações de servidores DNS e *Gateway*. O servidor DHCP fica escutando à rede em espera de solicitações DHCP por parte dos clientes que se conectam a ela (Vieira, 2010).

Neste modelo, o servidor possui as configurações de rede e, ao receber solicitação do cliente, envia estas para ele. Isso permite que um cliente, apto à utilização desse protocolo, que se conecte a esta rede receba automaticamente os parâmetros de rede como o endereço IP.

Existem três formas de configuração para o DHCP (Silva et al, 2011): Manual, automática e dinâmica.

Na primeira, manual, os parâmetros de configuração são atribuídos ao cliente de acordo com uma tabela que contém a configuração desejada por cada cliente e seu endereço MAC, se o endereço MAC do hospedeiro que está solicitando a configuração estiver na tabela o mesmo receberá do servidor as suas configurações desejadas. Assim, já existe uma configuração preestabelecida para cada cliente e o DHCP apenas transmite

essas configurações.

Já na configuração automática, apenas é atribuído ao cliente um endereço IP permanente qualquer, dentre uma faixa de endereços IPs válidos disponibilizados para serem distribuídos para os clientes da rede.

Na configuração dinâmica (Vieira, 2010) também é atribuído ao cliente um endereço IP de forma automática, mas existe a configuração de um limite determinado de tempo para que o hospedeiro possa continuar utilizando aquela configuração, tornando a mesma disponível para outro host no momento em que este limite de tempo chega ao fim.

Na configuração do DHCP é possível configurar o tempo em que o cliente pode permanecer com aquela configuração, no final do tempo estabelecido o servidor verifica se o cliente continua ativo. Caso o cliente solicite um tempo maior que o configurado nesta opção, é analisado o limite configurado que determina o tempo máximo que o cliente pode utilizar determinado IP.

O fato de ser a única maneira em que se permite atribuir de forma automática a outro cliente um endereço IP já recentemente utilizado faz com que a configuração dinâmica seja importante para atribuições de IP's para aqueles clientes que são conectados apenas de forma temporária na rede.

A rede pode utilizar mais de um desses tipos de DHCP se necessário, em alguns casos também deve ser reservada uma faixa de IP's para aqueles hosts que necessitarem de IP's fixos.

O grande benefício de se utilizar um servidor DHCP em uma rede é a própria automatização na atribuição de configurações de rede para os hospedeiros dessa rede, isso facilita todas essas configurações e as torna mais seguras já que evita erros que poderiam ocorrer em configurações feitas manualmente em cada host, principalmente em grandes redes. Com o servidor DHCP o administrador de rede precisa somente configurar no próprio servidor as configurações dos clientes, sem a necessidade de configuração manual em cada cliente (Marques, 2007).

2.2 TFTP

O *Trivial File Transfer Protocol* (TFTP) é um protocolo que proporciona transferência simples de arquivos entre máquinas, permitindo que arquivos de um servidor remoto possam ser enviados e recebidos por um cliente. A sua especificação encontra-se na RFC 350 (Sollins, 1992).

Por ser um protocolo simples, a transferência de arquivos pela rede é feita sem algum tipo de autenticação, nem verificação de erros ou algum outro mecanismo de segurança (Rodrigues, 2007). Assim, este protocolo utiliza o protocolo de camada de transporte UDP para seu funcionamento (Bezerra, 2007).

No caso de inicialização remota de clientes sem Sistema Operacional instalado, o TFTP é responsável por disponibilizar os arquivos de configuração e imagem para estes hospedeiros (Pereira, 2009).

2.3 PXE

O *Preboot Execution Environment* (PXE) é o protocolo que permite a inicialização de uma máquina através de sua interface de rede, independente de sua capacidade de armazenamento ou da existência de um Sistema Operacional instalado, padronizando a interação cliente/servidor para o *boot* remoto. Seu desenvolvimento se iniciou na Intel e é especificado através da RFC 4578.16 (Pires, 2009).

Para a comunicação entre o cliente e o servidor, o PXE utiliza os protocolos TCP/IP, DHCP e TFTP, além de definir as condições para interação entre esses protocolos sem que isso atrapalhe as características da rede (Achesinski et al, 2006).

Este protocolo especifica a comunicação pela qual um hospedeiro requisita e recebe uma imagem executável de um sistema, além de estabelecer os requisitos necessários ao cliente para que a imagem possa ser executada. Neste cenário, o cliente envia requisições *broadcast* até conseguir obter um endereço válido para acessar a rede.

Após o cliente receber os parâmetros de configurações de rede, ele recebe o programa *Network Bootstrap Program* (NBP) pelo servidor indicado pelo DHCP ou por um servidor de redirecionamento (servidor que fica escutando requisições na rede e encaminha

estas requisições para os servidores correspondentes). Este programa assume o controle das operações e é responsável por criar um ambiente padrão em todos os clientes daquele servidor, isso cria uma estrutura flexível e mais independente do cliente do que se este programa estivesse implementado diretamente no cliente e não recebido via PXE (Moreiral, 2008). O NBP também é responsável por, utilizando TFTP, carregar as configurações e o kernel do sistema (Quillan, 2006).

Depois ocorre uma série de configurações baseadas nas características do cliente e do servidor com a possibilidade de intervenção do usuário. Em seguida se dá o carregamento do Sistema Operacional.

Essa inicialização do Sistema Operacional no cliente pode ocorrer para reparação ou reinstalação de um Sistema Operacional já existente no cliente, para instalação do Sistema Operacional em um cliente com HD vazio ou simplesmente para iniciar um Sistema Operacional em um computador sem HD (Covelo et al, 2005).

Neste contexto, é importante que esteja configurado no servidor serviços que possibilitem o redirecionamento do cliente para um servidor de partida, ou seja, para um servidor que fornecerá a imagem ao cliente (Timoteo, 2010):

Para este redirecionamento, o servidor DHCP pode ser alterado de forma que, ao receber a solicitação do cliente, o servidor redirecione o mesmo para um servidor de partida desde que esteja sinalizada a requisição de *boot* remoto pelo cliente.

Também pode existir na infraestrutura da rede um servidor diferente do servidor DHCP, somente para redirecionamento desses clientes. Este servidor fica escutando na rede e responde a aqueles clientes que possuírem sinalização de PXE habilitada para realizar o redirecionamento para um servidor de partida.

2.4 PXELINUX

É um gerenciador de inicialização existente no NBP que permite obter um sistema operacional Linux de um servidor na rede.

Suporta o carregamento de um HD virtual e um núcleo de Sistema Operacional. A imagem de partida solicitada pela rede é carregada via TFTP (Hankins, 2007) e possui o formato pxelinux.0, esta imagem é carregada na memória RAM e executada.

2.5 NFS

O *Network File System* (NSF) é um sistema de arquivos distribuído, criado na década de 80 inicialmente pela Sun Microsystems e apoiado por outras empresas como IBM, Apple e Hewlett-Packard, que se tornou um dos sistemas mais utilizados para compartilhamento de arquivos através da rede nos ambientes UNIX (Ruppert, 2006).

Este sistema tem a intenção de proporcionar acesso remoto de forma transparente ao usuário utilizando a mesma semântica dos acessos locais, assim permite que programas em um terminal possam acessar arquivos remotos a ele como se estes fossem locais, permitindo que usuários em qualquer hospedeiro na rede possam acessar o sistema de arquivos que geralmente fica centralizado em um servidor na rede.

Os primeiros desses sistemas foram criados com a intenção de proporcionar o compartilhamento de dispositivos de armazenamento que na época possuíam altos preços.

Ele visa prover segurança e eficiência equivalente à que o sistema de arquivos teria se não fosse remoto e estivesse armazenado em discos locais (Barbosa et al, 2007), porém, o grande número de terminais que acessam o sistema e o grande tamanho de algumas redes contribuem para que nessas redes seja difícil garantir tais características.

O NFS adota a arquitetura cliente-servidor e esta organização também pode ser denominada de assimétrica. Porém, para diminuir o gargalo causado pela existência de apenas um servidor que é acessado por inúmeros clientes, também foi criado o NFSp.

Este sistema permite a existência de vários servidores de dados e um servidor de meta dados, que é enxergado pelos clientes e repassa as requisições feitas pelos mesmos para os servidores de dados existentes na estrutura. Também existe o dNFSp que permite a existência de mais de um servidor de meta dados, cada um responsável por receber as requisições de determinados grupos de clientes desse sistema (Boito, 2009), isso possibilita a divisão dos clientes entre esses servidores de meta dados evitando o ambiente com apenas um desses servidores atendendo a todos os clientes.

Ele é baseado no protocolo *Remote Procedure Call* (RPC) da Sun Microsystems e existem dois protocolos incluídos no NFS importantes para o funcionamento do mesmo. O primeiro é o Mount, através dele o cliente recebe um descritor de arquivos utilizado para referenciar os arquivos de forma remota e recebe a informação de quais sistemas de

arquivos estão disponíveis para montagem. Realizando assim a comunicação inicial entre cliente e servidor.

O outro protocolo leva o nome do sistema, NFS, e possibilita ao cliente modificar, incluir ou excluir arquivos, é ele que permite ao cliente acessar o conteúdo dos sistemas de arquivos (Ruppert, 2006).

2.6 XDMCP

O *X Display Manager Control Protocol* (XDMCP) se originou nas décadas de 70 e 80 com a sua utilização em terminais burros da época. Este protocolo é responsável por manipular os pedidos de inicio de sessão *X-Window* ao *X Display Manager* (XDM), responsável pela execução de sessão gráfica em um computador remoto.

Ele fornece uma forma de executar a aplicação gráfica no cliente, frequentemente nomeado de Terminal X, utilizando o servidor nomeado de Servidor X para fornecer uma interface entre os dispositivos de entrada e saída do cliente e a área de trabalho gerada pelo servidor (Chao, 2007). Assim são estabelecidos meios para a comunicação do usuário que está no cliente com o Servidor X, com essa comunicação estabelecida o usuário executa atividades e programas no cliente como se estivesse no servidor, o cliente neste caso é apenas uma janela do sistema rodando no servidor.

Por ser uma aplicação que funciona de forma cliente-servidor, o servidor X é capaz de rodar aplicativos em hospedeiros de toda a rede. Nesta estrutura, o cliente fica com uma carga de processamento mínima, já que a ele é atribuída apenas às funções de receber os dados via rede e exibir os mesmos, todo o restante do processamento e a grande parte do mesmo ficam por conta do servidor (Marimoto, 2006).

Esta tecnologia, além de permitir a exibição de sistemas em computadores leves sem Sistema Operacional e HD, também facilita o processo de administração dos computadores da rede. Afinal, com a adoção desse método se torna necessária somente a administração do ambiente gráfico do servidor, já que os ambientes dos clientes serão gerados pelo mesmo. Essas características representam uma maior facilidade para a criação de ambientes com vários clientes a partir da criação de um Servidor X com boa capacidade.

Por outro lado, a implementação dessa tecnologia requer a existência de uma rede

rápida, já que tudo que é realizado nos clientes depende da comunicação pela rede entre esses e o servidor (Ferretti, 2004).

3 Linux Terminal Server Project - LTSP

O *Linux Terminal Server Project* (LTSP), foi criado em 1996 por James McQuillan, nos Estados Unidos (Souza et al, 2012). Este projeto de código livre hoje tem o apoio de grande comunidade no mundo todo, sendo que o Brasil é um dos países com maior número de implementações baseadas neste projeto, inclusive em órgãos governamentais (Ferretti, 2004).

O LTSP tende a reunir ferramentas que permitam que terminais chamados de clientes leves operem utilizando recursos computacionais de servidores, a partir da comunicação desses pela rede. Assim, fica a cargo desses clientes apenas a exibição das imagens recebidas do servidor e o envio para o servidor das solicitações e escolhas do usuário (Michelin et al, 2007). Estes clientes não precisam possuir disco rígido, Sistema Operacional e nem boa capacidade de memória e processamento.

Com a estrutura LTSP construída, o cliente inicia e envia via broadcast uma solicitação DHCP na rede LTSP em que ele está. O sistema inetd, responsável por aguardar requisições de conexão de rede, recebe a solicitação e chama o DHCP para tratá-la. O serviço DHCP, configurado em algum servidor da estrutura de rede, disponibiliza um IP ao cliente e a localização do servidor onde está o NBP.

Dentro desse programa de boot está o PXELINUX, ele vai permitir ao cliente receber via TFTP a imagem do Sistema Operacional e carrega-la, esta imagem possui o formato 'pxelinux.0'. Após receber a imagem do sistema, uma série de configurações é realizada no mesmo.

Em seguida, é montado o sistema de arquivos que será utilizado pelo terminal através do NFS.

Com o cliente inicializado, são realizadas algumas configurações no arquivo lts.conf (Santos et al, 2012) do servidor, configurações de resolução de vídeo, ativação de *swap*, tipo de mouse e tipo de teclado.

Finalmente, o servidor X fornece ao cliente a sua interface gráfica.

Este processo é ilustrado na Figura 3.1.

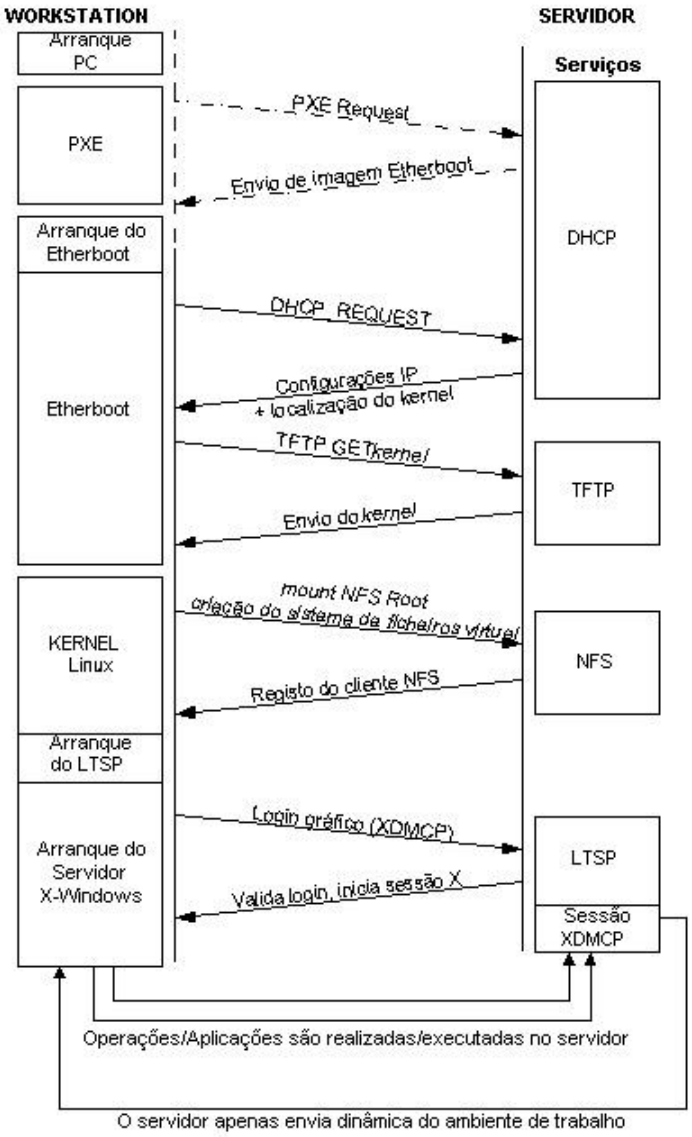


Figura 3.1: Processo de inicialização do cliente LTSP (Santos et al, 2012)

Podemos considerar como sendo vantagem do LTSP a facilidade na manutenção e gerenciamento dos terminais. Uma vez que os aplicativos estão instalados todos no servidor, configurações nos softwares, atualizações e instalações precisam ser realizadas apenas no servidor e não em cada terminal. A rede externa é compartilhada automaticamente via DHCP com todos os terminais, bastando para isso configurar este serviço também somente no servidor.

A partir da criação de um *proxy* em um dos servidores é possível fornecer acesso dos terminais à rede externa bloqueando o acesso a sites não permitidos pela política da instituição em que a arquitetura LTSP está implementada. Com um *firewall* neste mesmo

servidor, pode-se constituir uma maior segurança para a rede interna do LTSP (Ramos, 2012).

Também é vantagem da tecnologia LTSP o custo de implementação, devido ao fato de poder utilizar terminais sem HD e com baixa capacidade de processamento. O investimento se concentra mais na estrutura de rede e no servidor, esse custo fica menor do que em uma rede onde todas as máquinas teriam que possuir bom desempenho. Também contribui para a diminuição do custo o fato de no LTSP não ser necessário o gasto com licenças de *software*, já que é feito tudo baseado em *softwares* livres.

A centralização no servidor também gera algumas desvantagens da utilização do LTSP, por todos os terminais dependerem do servidor, se este falhar, todos os clientes ficarão inoperantes ou também falharão. (Ramos, 2012) Todos os arquivos e configurações dos usuários dos terminais clientes estão também centralizados em servidor, algum problema com o(s) servidor(es) pode acarretar na perda das configurações de todos os usuários dos terminais. Além disso, caso o servidor ou a rede não possua um bom desempenho, nenhum dos terminais possuirão.

Outra desvantagem dessa estrutura é que se um usuário está executando algum processo que consuma muito recurso de rede e de servidor, as solicitações feitas pelo terminal desse usuário ao servidor poderão sobrecarregar o mesmo e a rede atrapalhando as execuções dos usuários que estão nos outros terminais.

3.1 LTSP-Cluster

O *LTSP-Cluster* é uma solução que reúne diversas ferramentas e a solução LTSP original. Esta solução permite contornar limites de desempenho que ocorrem na utilização de um único servidor e proporcionar que mais de um servidor de aplicação opere na rede LTSP.

Para isso, ele permite a divisão da carga total gerada pelos clientes entre esses servidores, isso facilita o aumento do número de clientes já que flexibiliza o aumento da capacidade dos servidores, de forma que na necessidade de maior capacidade basta incluir uma nova máquina no *cluster*.

Uma dessas ferramentas incorporadas pelo *LTSP-Cluster* é o *Load Balancer*, este mecanismo de distribuição de carga recebe as requisições feitas aos servidores e divide as

mesmas entregando-as de forma balanceada a cada um dos servidores (Luis, 2008). Como é este o mecanismo que coordena e distribui as solicitações realizadas ao servidor, é de suma importância seu funcionamento para o funcionamento do *TCOS-Cluster* e um dos pontos críticos dessa arquitetura.

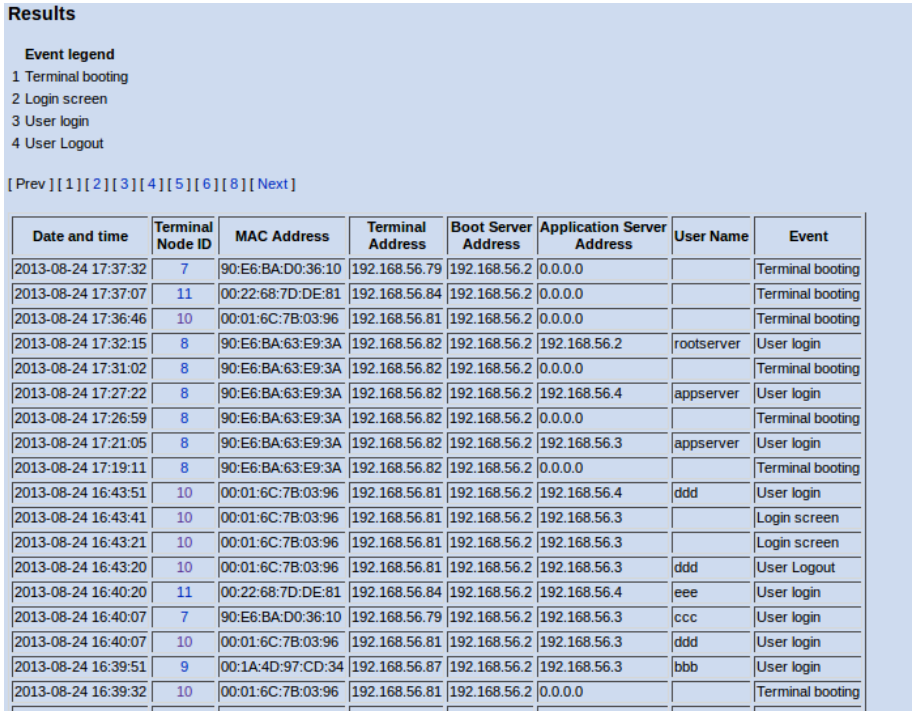
O *Load Balancer* possui um arquivo de configuração em que são informados os IP's dos servidores que estão no *cluster* LTSP para que possam participar do balanceamento de carga do *Load Balancer*.

A documentação do *LTSP-Cluster-Control* também cita o agente *Load Balancer* para Microsoft Windows, que permite mesclar servidores Linux e Windows na estrutura LTSP. Porém, este trabalho não se aprofunda nesta possibilidade por esta fugir do escopo do trabalho, que é o de não trabalhar com tecnologia proprietária.

O Centro de Controle *LTSP-Cluster-Control* é outra ferramenta que compõe o *LTSP-Cluster*. Esta ferramenta de interface *web* permite configurações e gerenciamento do *cluster* e dos clientes. Entre essas configurações estão: O endereço dos servidores (o caminho para o *Load Balancer*, neste caso), o endereço do servidor de data e hora (*TIMESERVER*), um inventário dos clientes que acessam o servidor e os *logs* dos mesmos. A Figura 3.2 mostra a tela onde são exibidos os *logs* de cada cliente no sistema LTSP, nela estão as identificações dos clientes no sistema, informações de IP e MAC dos mesmos, estado em que cada cliente está e horário da atualização desse estado.

3.2 Rdesktop

O LTSP tem a opção da utilização do *Rdesktop*, um cliente *Open Source*, para iniciar uma conexão *Remote Desktop Protocol* (RDP). Isso, por exemplo, pode ser também uma opção para acessar um servidor *Windows* no cliente LTSP e possibilitar aos clientes o acesso a diversos servidores RDP (Tsai et al, 2011). Para isso, o *Rdesktop* deve ser instalado no *chroot* (imagem do sistema operacional disponibilizada para o cliente) do cliente.



Results

Event legend
1 Terminal booting
2 Login screen
3 User login
4 User Logout

[Prev] [1] [2] [3] [4] [5] [6] [8] [Next]

Date and time	Terminal Node ID	MAC Address	Terminal Address	Boot Server Address	Application Server Address	User Name	Event
2013-08-24 17:37:32	7	90:E6:BA:D0:36:10	192.168.56.79	192.168.56.2	0.0.0.0		Terminal booting
2013-08-24 17:37:07	11	00:22:68:7D:DE:81	192.168.56.84	192.168.56.2	0.0.0.0		Terminal booting
2013-08-24 17:36:46	10	00:01:6C:7B:03:96	192.168.56.81	192.168.56.2	0.0.0.0		Terminal booting
2013-08-24 17:32:15	8	90:E6:BA:63:E9:3A	192.168.56.82	192.168.56.2	192.168.56.2	rootserver	User login
2013-08-24 17:31:02	8	90:E6:BA:63:E9:3A	192.168.56.82	192.168.56.2	0.0.0.0		Terminal booting
2013-08-24 17:27:22	8	90:E6:BA:63:E9:3A	192.168.56.82	192.168.56.2	192.168.56.4	appserver	User login
2013-08-24 17:26:59	8	90:E6:BA:63:E9:3A	192.168.56.82	192.168.56.2	0.0.0.0		Terminal booting
2013-08-24 17:21:05	8	90:E6:BA:63:E9:3A	192.168.56.82	192.168.56.2	192.168.56.3	appserver	User login
2013-08-24 17:19:11	8	90:E6:BA:63:E9:3A	192.168.56.82	192.168.56.2	0.0.0.0		Terminal booting
2013-08-24 16:43:51	10	00:01:6C:7B:03:96	192.168.56.81	192.168.56.2	192.168.56.4	ddd	User login
2013-08-24 16:43:41	10	00:01:6C:7B:03:96	192.168.56.81	192.168.56.2	192.168.56.3		Login screen
2013-08-24 16:43:21	10	00:01:6C:7B:03:96	192.168.56.81	192.168.56.2	192.168.56.3		Login screen
2013-08-24 16:43:20	10	00:01:6C:7B:03:96	192.168.56.81	192.168.56.2	192.168.56.3	ddd	User Logout
2013-08-24 16:40:20	11	00:22:68:7D:DE:81	192.168.56.84	192.168.56.2	192.168.56.4	eee	User login
2013-08-24 16:40:07	7	90:E6:BA:D0:36:10	192.168.56.79	192.168.56.2	192.168.56.3	ccc	User login
2013-08-24 16:40:07	10	00:01:6C:7B:03:96	192.168.56.81	192.168.56.2	192.168.56.3	ddd	User login
2013-08-24 16:39:51	9	00:1A:4D:97:CD:34	192.168.56.87	192.168.56.2	192.168.56.3	bbb	User login
2013-08-24 16:39:32	10	00:01:6C:7B:03:96	192.168.56.81	192.168.56.2	0.0.0.0		Terminal booting

Figura 3.2: Tela de *Logs* do *LTSP-Cluster-Control* utilizado no NRC

3.3 Epoptes

Esta ferramenta de monitoramento *Open Source* voltada para laboratórios de ensino, segundo o site de seus desenvolvedores e mantenedores, permite o monitoramento de terminais remotos, estações de trabalho independentes ou clientes LTSP.

Seu monitoramento proporciona: A visualização das telas dos clientes; a execução remota de comandos nas máquinas dos clientes; o bloqueio de telas; o bloqueio do terminal depois de determinado tempo; definir configurações do cliente; transmitir uma tela ao cliente (a tela do gerenciador, por exemplo); receber ou enviar arquivos remotamente; enviar mensagens aos clientes e aplicação de políticas de restrições como, por exemplo, a diminuição do som de um cliente. A Figura 3.3 mostra uma tela do Epoptes, nela é ilustrado o monitoramento das telas de um grupo de clientes e a possibilidade de realizar ações para todos aqueles clientes.

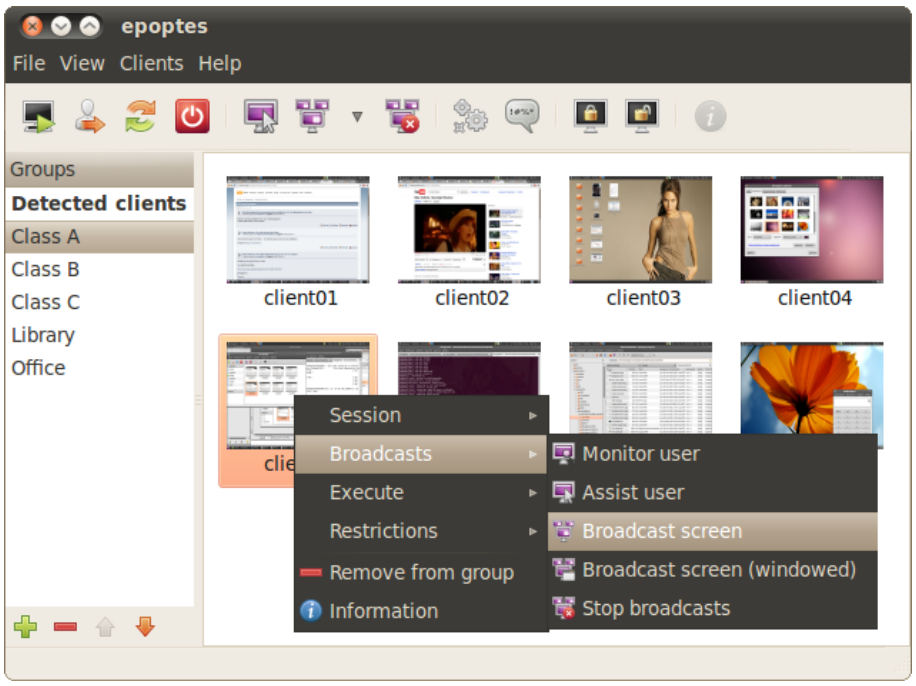


Figura 3.3: Tela da ferramenta Epoptes

3.4 Conclusão

Neste capítulo foram apresentados os principais conceitos do LTSP, seu funcionamento e as principais ferramentas que possuem integração com o mesmo.

Estes conceitos são importantes para o entendimento dos testes realizados já que, para analisar os mesmos no próximo capítulo, é importante entender o funcionamento do LTSP e as ferramentas utilizadas na construção do ambiente de testes no NRC.

4 Análise de Desempenho

Os testes realizados neste trabalho objetivaram coletar dados de tráfego em rede, consumo de CPU e consumo de memória. Foram realizados testes utilizando os três servidores da nossa estrutura, *rootserver*, *appserver* e *appserver2*, todos eles descritos neste capítulo. Também foram utilizadas oito máquinas (também descritas neste capítulo) do Núcleo de Recursos Computacionais do ICE para que servissem de clientes LTSP.

4.1 Ambiente de Realização

Toda a realização dos mesmos se deu no Núcleo de Recursos Computacionais do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora. A estrutura criada no NRC para a realização dos testes é ilustrada na Figura 4.1.

Foram criados três servidores LTSP. O *rootserver* foi o primeiro servidor a ser criado.

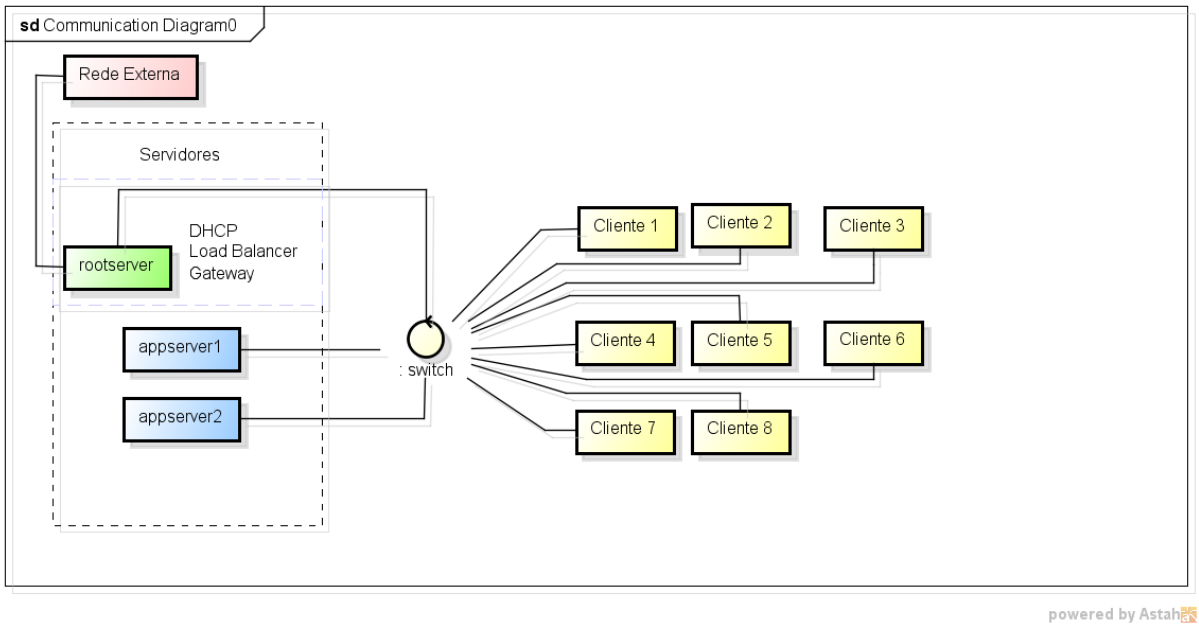


Figura 4.1: Estrutura criada para a realização dos testes

É no *rootserver* que está configurado o serviço DHCP fazendo com que este atue

como o servidor DHCP da rede, também é nele que a imagem, que os clientes carregarão no momento do *boot*, é compilada e através dele fornecida aos clientes.

O *Load Balancer* também está configurado no *rootserver*, assim é este servidor o centro de administração da estrutura de *Cluster* do LTSP.

Este servidor também foi configurado para ser o *gateway* da rede LTSP, permitindo acesso dos clientes à internet. Possui duas interfaces de rede, uma é responsável pelo acesso à rede externa (rede do NRC) e outra responsável pela comunicação com as maquinas da rede interna, ou seja, comunicação com clientes e servidores de aplicação.

Os outros dois servidores criados são os servidores de aplicação que a tecnologia *LTSP-Cluster* permite adicionar à estrutura. São neles que os dados dos clientes são processados após o *rootserver* fornecer a imagem do Sistema Operacional ao cliente e o *Load Balancer* indicar um desses dois servidores para responder às solicitações dos mesmos.

Tanto no *rootserver* quanto nos dois servidores de aplicação, foi instalado o Sistema Operacional Ubuntu 12.04 com a interface Gnome.

Cada um desses dois servidores de aplicação possui 2 GB de memória e 2 CPU's Intel Core i7 de 3.4 GHz de frequência.

As máquinas utilizadas como clientes LTSP possuem as seguintes configurações:

Cliente 1: Processador AMD Athlon II X2 245 de 2,9 GHz de frequência. Memória de 4096 MB.

Cliente 2: Processador Intel Core 2 Quad Q8400 de 2,66 GHz de frequência. Memória de 4096 MB.

Cliente 3: Processador Intel Core 2 Quad Q8400 de 2,66 GHz de frequência. Memória de 4096 MB.

Cliente 4: Processador Intel Pentium 4 de 3,00 GHz de frequência. Memória de 4096 MB.

Cliente 5: Processador Intel Core 2 Quad Q8400 de 2,66 GHz de frequência. Memória de 4096 MB.

Cliente 6: Processador Intel Core i3 2100 de 3,10 GHz de frequência. Memória de 4096 MB.

Cliente 7: Processador Intel Core i3 2100 de 3,10 GHz de frequência. Memória de 4096 MB.

Cliente 8: Processador Intel Core i3 2100 de 3,10 GHz de frequência. Memória de 4096 MB.

Todas as placas de rede dos 3 servidores utilizadas para comunicação na rede LTSP e as placas de rede de 6 clientes são de 1000Mbps, sendo que 2 clientes possuem placa de rede de 100 Mbps. Porém, as placas de rede do *switch* utilizado para estabelecer comunicação entre os servidores e os clientes são de 100 Mbps, o que limita a velocidade da comunicação na rede LTSP em 100 Mbps.

Na análise dos resultados obtidos através dos testes, é importante saber que, como um servidor possui dois processadores, o máximo de processamento possível em um servidor é dado como 200%. Em relação à memória, o máximo de capacidade (os 2 GB) de memória existentes em cada servidor é dado como 100%.

Nos resultados dos testes com dois servidores, foi realizada a média para um servidor. Assim, os gráficos exibem a média de consumo de apenas um servidor, ou seja, a metade do consumo total da estrutura. Também é representado o máximo de CPU em cada servidor como 200% e o máximo de memória em cada servidor como 100%.

4.2 Cenários

O processo de realização dos testes ocorreu dividido em alguns cenários.

O primeiro desses cenários teve o objetivo de analisar o tráfego na interface de rede dos servidores em um de seus momentos mais críticos, no momento da inicialização do sistema. O segundo cenário é o que analisa os consumos de CPU e memória da aplicação Firefox quando executada no ambiente LTSP. O último cenário também objetiva analisar os consumos de CPU e memória, porém é analisada a aplicação NetBeans. Estes cenários, as justificativas e objetivos das escolhas dos mesmos, os testes realizados com eles e seus resultados, são descritos neste capítulo.

4.2.1 Análise de tráfego na inicialização

Este cenário possuiu o objetivo de coletar e analisar o tráfego nas interfaces de rede dos servidores da estrutura, principalmente do *rootserver*, que é quem recebe as solicitações de todos os clientes e fornece a cada um dos mesmos a sua configuração de rede (através do DHCP) e sua imagem de inicialização (via PXE).

Neste teste foi executado nos servidores um *script* que realiza 80 consultas sucessivas à interface de rede do servidor e coleta a variação do tráfego de entrada e saída dessa interface em todas estas consultas. As variações do tráfego de entrada e do tráfego de saída são somadas e aparecem como uma única variação nos resultados desses testes.

Com isso é possível analisar o comportamento da rede em todo o processo de inicialização dos clientes.

O *script* foi executado em um total de nove vezes, três vezes com dois clientes na rede LTSP, três vezes com quatro clientes na rede LTSP e mais três vezes com oito clientes nesta rede. Em todas as vezes que era executado, iniciava a sua execução antes do cliente começar a sua inicialização e terminava após o cliente estar inicializado e o *login* de algum usuário realizado no mesmo.

Começando a analisar os resultados obtidos a partir das coletas de dados e resumidos nestes gráficos, podemos observar o gráfico da Figura 4.2. Este gráfico mostra o tráfego de dados na interface de rede do servidor *rootserver* no momento da inicialização dos clientes. Podemos notar que o tráfego em todos os casos, principalmente com quatro e oito clientes, obteve duas elevações mais duradouras, iniciando na iteração sete, onde os clientes começavam as suas inicializações.

O que ocorre na primeira elevação é que os terminais estão se comunicando com o servidor DHCP para obterem as suas configurações de rede e o endereço de um servidor de partida. Após o cliente receber estas configurações, ele realiza algumas configurações locais de rede e carrega o NBP, o que é responsável por grande parte da primeira elevação no tráfego. Após carregar o NBP ocorre a execução do mesmo localmente, isto explica a diminuição de comunicação do cliente com o servidor e a queda do tráfego no gráfico. Essa queda ocorre na iteração 14 quando executados dois clientes, na iteração 15 quando executados quatro clientes e na iteração 27 quando executados oito clientes.

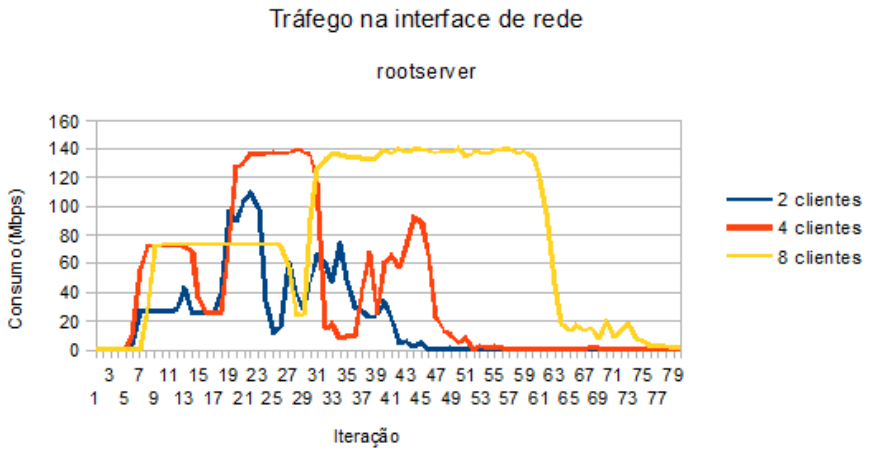


Figura 4.2: Tráfego na interface de rede do servidor *rootserver*

Após criar no cliente um ambiente propício para carregar a imagem do Sistema Operacional, o NBP inicia o processo de carregar a imagem do Sistema Operacional do cliente, isso atinge o ápice da comunicação entre o cliente e o servidor e é responsável pelo aumento de maior duração do tráfego na interface do servidor para cada um dos casos (dois, quatro e oito clientes), demonstrado no gráfico.

Também podemos observar que os picos de tráfego na interface do servidor quando oito máquinas estão iniciando dura mais que no momento em que duas ou quatro máquinas estão iniciando.

Isso mostra que quando existem oito clientes iniciando simultaneamente a inicialização desses clientes demora mais para ser completada do que quando existem menos clientes. No gráfico podemos observar que, quando iniciados quatro clientes, na iteração 50 os seus Sistemas Operacionais já estavam carregados. Mas, quando iniciados oito clientes, somente na iteração 67 os seus Sistemas Operacionais já estavam carregados.

Após o sistema do cliente ser carregado, através do *Load Balancer* o processamento do cliente é passado do *rootserver* para o servidor de aplicação. Se analisarmos a Figura 4.3 do gráfico que nos mostra o tráfego no *rootserver* e no servidor de aplicação enquanto oito clientes são iniciados na rede, notamos que o tráfego na interface de rede de um dos servidores de aplicação começa no momento em que termina o tráfego na interface do *rootserver*.

Notamos alguns picos no tráfego do servidor de aplicação, estes picos ocorrem no

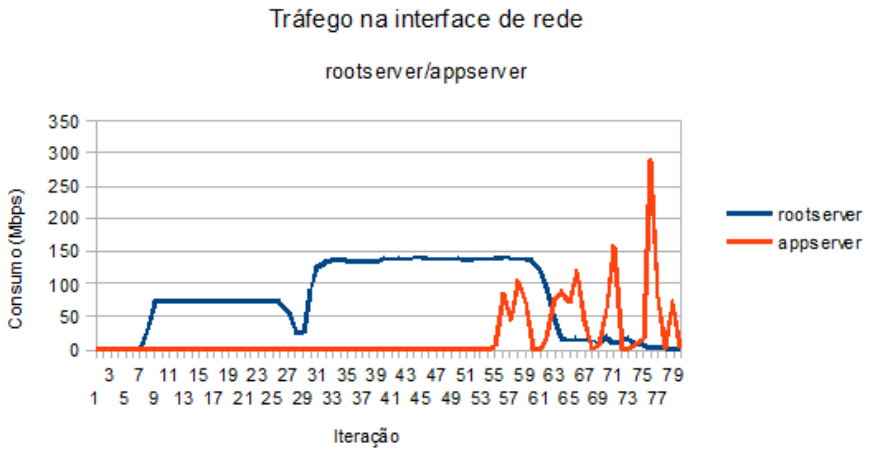


Figura 4.3: Tráfego nas interfaces de rede do servidor *rootserver* e do servidor de aplicação *appserver*

momento em que são realizados os *logins* dos usuários no sistema e o carregamento de algumas configurações dos mesmos.

Os testes também nos mostraram, com o auxilio dos gráficos, que no momento da inicialização simultânea de quatro e de oito máquinas o tráfego quase chegou à casa dos 140 Mbps no rootserver e 280 Mbps no servidor de aplicação.

Neste cenário, também foi possível observar o funcionamento do *Load Balancer* na distribuição de carga entre os servidores.

Ao iniciar apenas uma máquina com os dois servidores de aplicação habilitados, foi possível confirmar que o processamento inteiro do cliente se deu em apenas um servidor, não distribuindo a carga de um mesmo cliente entre dois servidores.

Ao executar mais de um cliente podemos confirmar que o *Load Balancer*, apesar de propor distribuir igualmente a quantidade de clientes para os servidores, não realizou nos testes a distribuição dessa forma. Nas três inicializações com oito clientes e dois servidores de aplicação habilitados, em duas vezes o *Load Balancer* atribuiu seis clientes para um servidor de aplicação e dois clientes para outro servidor (sendo que no primeiro teste foi o *appserver1* quem recebeu seis clientes e no segundo teste foi o *appserver2* quem recebeu os seis clientes). No terceiro teste o *Load Balancer* atribuiu cinco clientes ao *appserver1* e três clientes ao *appserver2*.

4.2.2 Consumo de CPU e memória executando Firefox

Este cenário teve o objetivo de verificar o comportamento de CPU e memória dos servidores na execução do sistema de navegação web Firefox. A proposta desse cenário era coletar e analisar dados que pudessem nos dar informações de como o LTSP se comporta na utilização de seus clientes para tarefas simples, como acesso a internet através de um navegador web, tarefa esta comumente realizada em centros de informática voltados para a população, por exemplo.

Para isso, foi utilizado um *script* que realizava 80 iterações e a cada iteração realizava consultas que retornavam as porcentagens de CPU e memória utilizadas naquele momento pelo processo responsável pela execução do Firefox.

Este *script* foi executado três vezes para a análise de dois clientes conectados à rede LTSP, três vezes para quatro clientes executando nesta rede e mais três vezes para oito clientes executando na mesma rede.

A cada execução do *script*, todos os clientes envolvidos naquele teste tinham o Firefox iniciado (com o acesso ao site YouTube) simultaneamente e, na iteração 40, eles executavam um vídeo no mesmo (o mesmo vídeo para todos os clientes).

O *script* era executado no momento em que nenhum cliente estava com o Firefox aberto, contudo o mesmo ficava aguardando todos os clientes estarem com algum processo do Firefox executando para começar a primeira iteração de coleta dos dados.

Todos estes processos eram realizados com apenas um servidor de aplicação habilitado no *Load Balancer*, sendo este o único responsável pelas execuções dos clientes.

Em um segundo momento, o segundo servidor de aplicação também era habilitado, fazendo com que as cargas requisitadas pelos clientes fossem distribuídas entre os mesmos. Todo o processo de teste citado acima se repetia com os dois servidores habilitados.

Executando o Firefox, podemos notar que os servidores se comportaram muito bem e que o LTSP funciona bem com sistemas leves como um navegador web.

Analisando a Figura 4.4 do gráfico em que mostra o consumo de CPU total dos clientes quando estes estão enviando requisições apenas para um servidor, podemos observar que o total do consumo não chega nem a 40% da capacidade do servidor. Mesmo quando

as aplicações são iniciadas e quando é executado algum vídeo no YouTube. Também podemos observar que, a partir da iteração 40, com o inicio da execução dos vídeos, o consumo de CPU aumenta em relação ao momento em que o vídeo não está sendo executado.

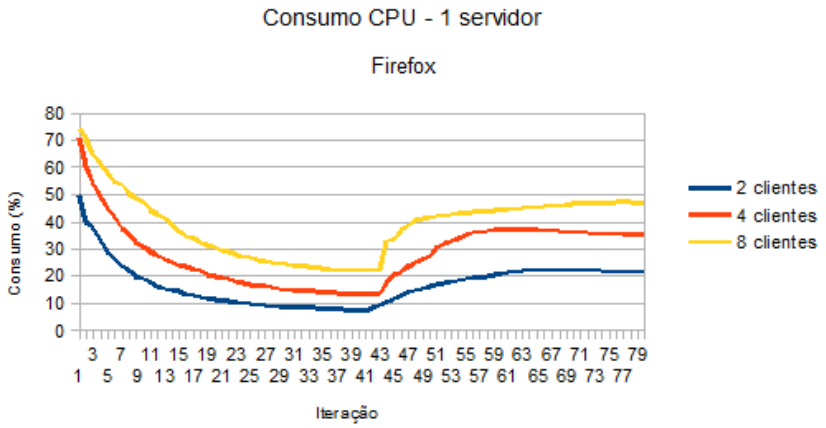


Figura 4.4: Consumo total de CPU executando Firefox com um servidor de aplicação

Quando se trata de consumo de memória, o mesmo requer uma porcentagem maior da capacidade do servidor, principalmente com oito clientes. Quatro clientes, quando executados, não ultrapassaram 40% da capacidade total do servidor, mas oito clientes atingiram quase 70% do potencial de memória do servidor, como podemos observar na Figura 4.5. Neste gráfico também podemos perceber que, assim como ocorre com a CPU, o consumo de memória aumenta com a execução de vídeos.

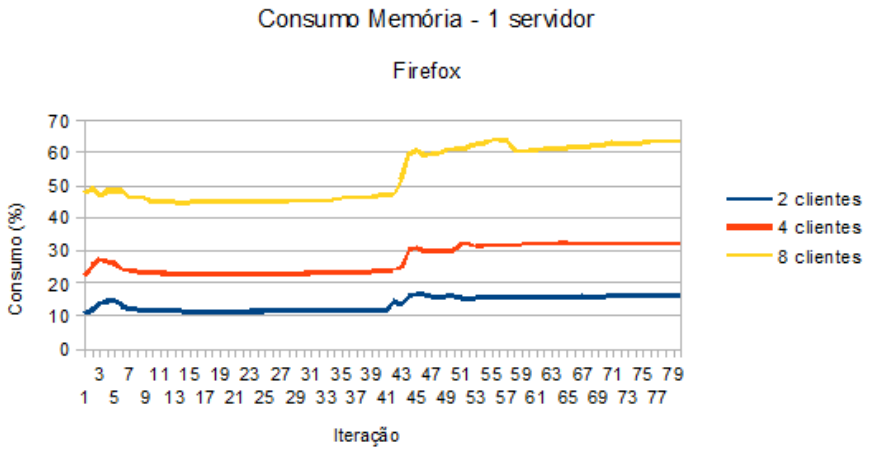


Figura 4.5: Consumo total de memória executando Firefox com um servidor de aplicação

Ao observarmos os gráficos que mostram a média de consumo para cada cliente,

tanto de CPU quanto de memória, conseguimos ver que quanto mais clientes existem para aquele servidor, menor é a porcentagem de recursos que este cliente conseguirá obter. Mesmo a porcentagem total de gasto dos recursos do servidor sendo maior quando existem mais clientes dependendo deste servidor. Os gráficos que confirmam isto são os exibidos nas Figuras 4.6 e 4.7.

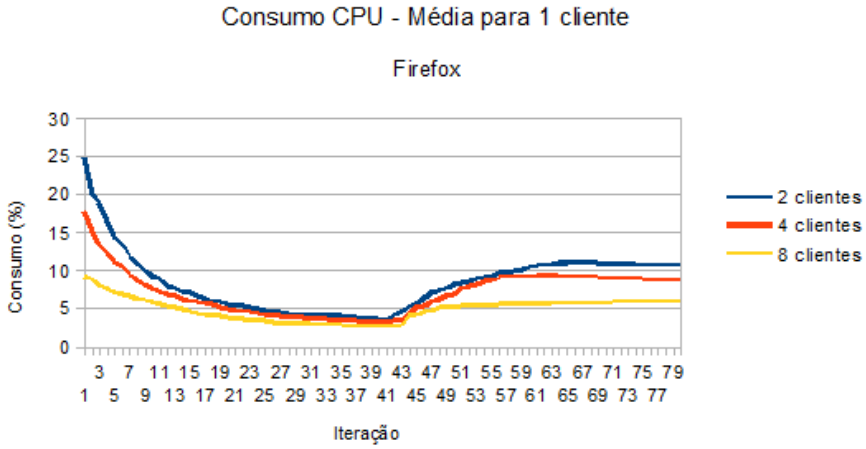


Figura 4.6: Consumo médio de CPU para um cliente executando Firefox

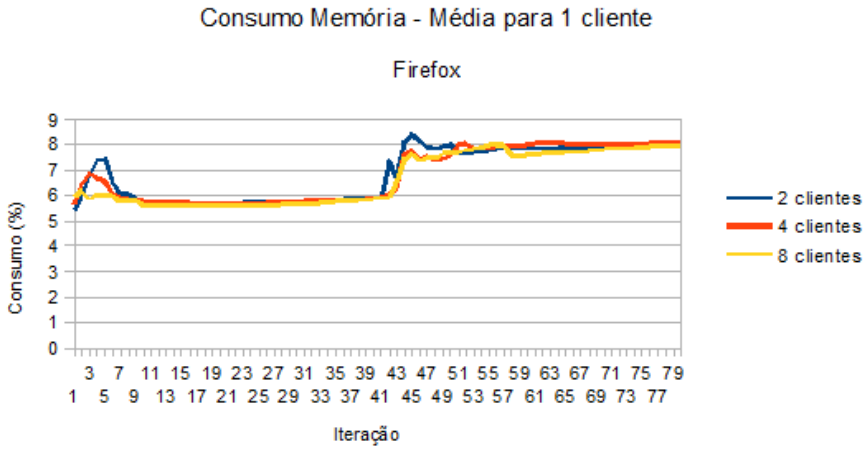


Figura 4.7: Consumo médio de memória para um cliente executando Firefox

Como o consumo gerado pelas aplicações não chegou ao limite de capacidade de um servidor, este consumo não apresentou diferença quando adicionado mais um servidor na rede, a média de consumo de cada um dos servidores é a metade do consumo do servidor quando este era o único servidor de aplicação na estrutura. Isso pode ser confirmado nos

gráficos das Figuras 4.8 e 4.9 que mostram a média de consumo de cada servidor quando existem dois servidores de aplicação na estrutura LTSP.

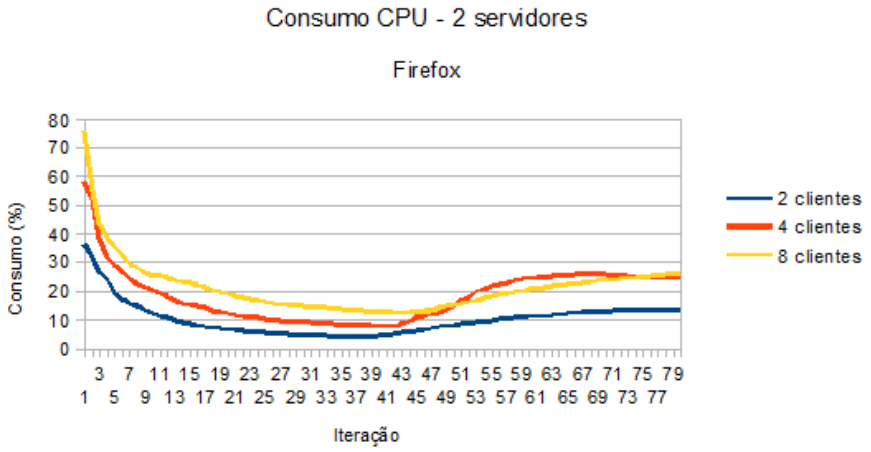


Figura 4.8: Consumo total de CPU executando Firefox com dois servidores de aplicação

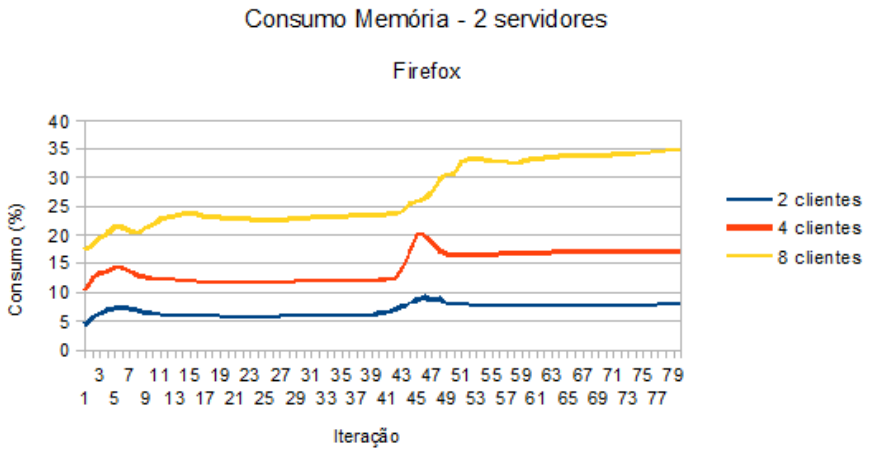


Figura 4.9: Consumo total de memória executando Firefox com dois servidores de aplicação

4.2.3 Consumo de CPU e memória executando NetBeans

Ao contrário do cenário anterior, este cenário propunha recolher dados que permitissem analisar o comportamento do LTSP com o emprego de seus clientes em tarefas mais pesadas, como a edição e compilação de um código Java através da ferramenta NetBeans. Isso permite analisar o desempenho atingido pelo LTSP quando este é empregado em laboratórios que requerem maior desempenho do que o necessário em um centro de informática

voltado para a comunidade. Um exemplo desses laboratórios pode ser um laboratório de programação onde é utilizado o NetBeans. Assim é possível analisarmos a possibilidade do emprego do LTSP até mesmo em laboratórios de universidades.

Para a coleta de dados desse cenário foi utilizado o mesmo *script* do cenário anterior. Porém, em cada uma das 80 iterações realizadas pelo *script*, eram realizadas consultas das porcentagens utilizadas de CPU e memória do processo responsável pela execução do ambiente de desenvolvimento em Java NetBeans.

O *script* também foi executado em um total de nove vezes, com três execuções para dois clientes, para quatro clientes e para oito clientes.

O teste também era iniciado antes que os clientes estivessem com o NetBeans iniciado e o *script* ficava aguardando que todos os clientes possuíssem algum processo relacionado ao NetBeans. Na iteração 40 era compilado em cada cliente o código do projeto Java que estava aberto no NetBeans de cada um dos mesmos.

O código editado e compilado era o mesmo em cada cliente, um editor de textos que vem como exemplo de implementação Java no sistema NetBeans.

Porém, a análise do NetBeans não se tratava apenas de um processo responsável pela execução da aplicação em cada cliente, ao compilar o código Java em cada NetBeans, um novo processo era gerado pela execução da Máquina Virtual Java responsável pela compilação daquele código. A partir desse momento, dois processos passavam a ser monitorados pelo *script* em cada cliente. Para resultado desse monitoramento as porcentagens de memória e CPU dos dois processos de cada cliente foram somadas e originou valores que representam a porcentagem total de CPU e a porcentagem total de memória requisitadas pelo cliente ao servidor para a execução do NetBeans.

Todos estes processos eram realizados com apenas um servidor de aplicação habilitado no *Load Balancer*, sendo este o único responsável pelas execuções dos clientes. Em um segundo momento, o segundo servidor de aplicação era habilitado e dados de desempenho eram gerados analisando o desempenho dos dois servidores para os dois, quatro e cinco clientes, como ocorreu com o Firefox.

Notamos pelos resultados que, ao executar o NetBeans, a capacidade exigida do servidor foi bem maior.

A partir dos gráficos podemos notar que existem dois picos de utilização de CPU pelo NetBeans, o primeiro é quando este sistema é iniciado e o segundo é quando ele é utilizado para compilar um código Java.

Observando o gráfico da Figura 4.10, onde mostra a capacidade de CPU de um servidor consumida pelos clientes, podemos constatar que com quatro clientes o servidor chegou a quase atingir a capacidade máxima dos seus dois processadores.

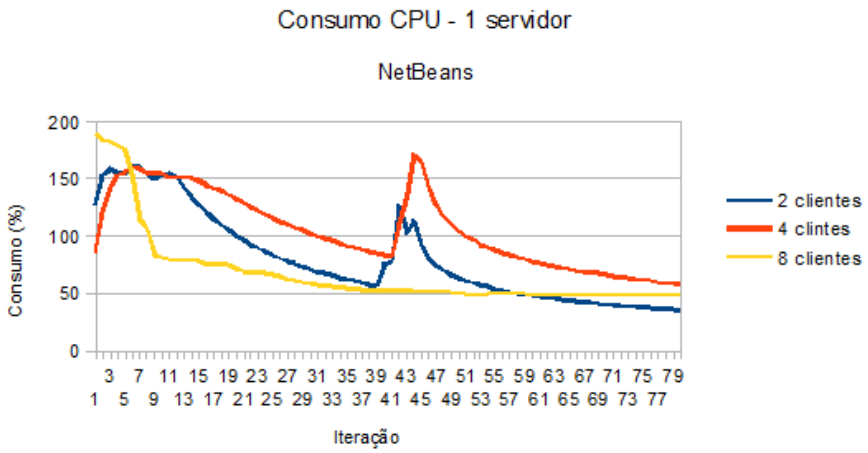


Figura 4.10: Consumo total de CPU executando NetBeans com um servidor de aplicação

Até mesmo quando executados apenas dois clientes, estes chegaram a consumir a capacidade total de um processador mais a metade da capacidade total do outro processador, no momento da inicialização do NetBeans.

Porém, o momento mais crítico ocorreu na execução de oito clientes. Neste caso o *script* foi executado apenas uma vez já que, no momento da inicialização dos oito clientes, o consumo de CPU da aplicação atingiu o limite do servidor e fez com que este parasse de responder à rede LTSP. Os clientes foram desligados e o servidor teve que ser reiniciado. No gráfico da Figura 4.10 podemos observar este pico no inicio do teste, onde ocorreu a queda do servidor.

O consumo de memória também foi grande executando o NetBeans, observando a Figura 4.11, onde mostra o consumo de memória de um servidor para a aplicação NetBeans, podemos notar que este consumo chegou a quase totalidade da capacidade do servidor quando executados quatro clientes. Ao serem executados dois clientes o consumo chegou quase à metade da capacidade total do servidor.

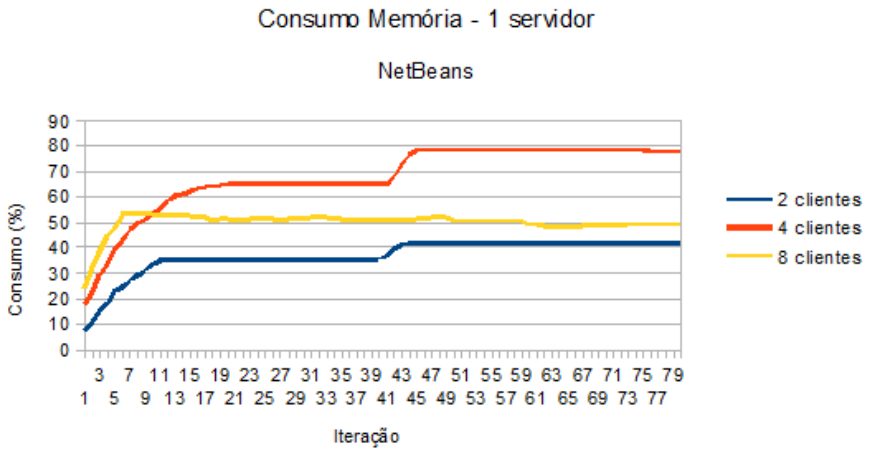


Figura 4.11: Consumo total de memória executando NetBeans com um servidor de aplicação

Como o servidor parou de responder aos clientes por causa da grande capacidade de CPU exigida pelos oito clientes, o consumo de memória quando executados estes clientes obteve uma elevação no começo da execução dos testes mas, graças à perda de comunicação com o servidor, não mais foi informado, se mantendo constante no gráfico.

Também podemos notar neste gráfico que o maior consumo de memória ocorre depois que é compilado o código Java na ferramenta, a partir da iteração 40.

Aquilo que ocorre na execução do Firefox também ocorre com o NetBeans, podemos confirmar que maior é o desempenho do cliente se menor for a quantidade de clientes na rede, isso pode ser visto nos gráficos das Figuras 4.12 e 4.13.

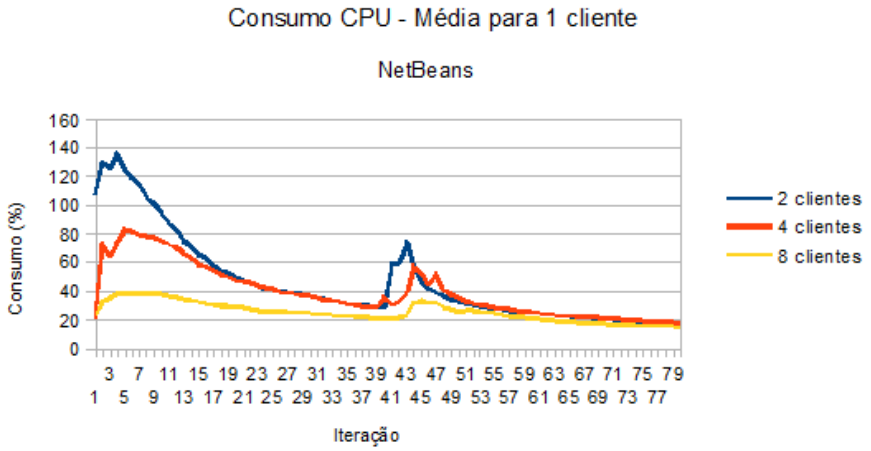


Figura 4.12: Consumo médio de CPU para um cliente executando NetBeans

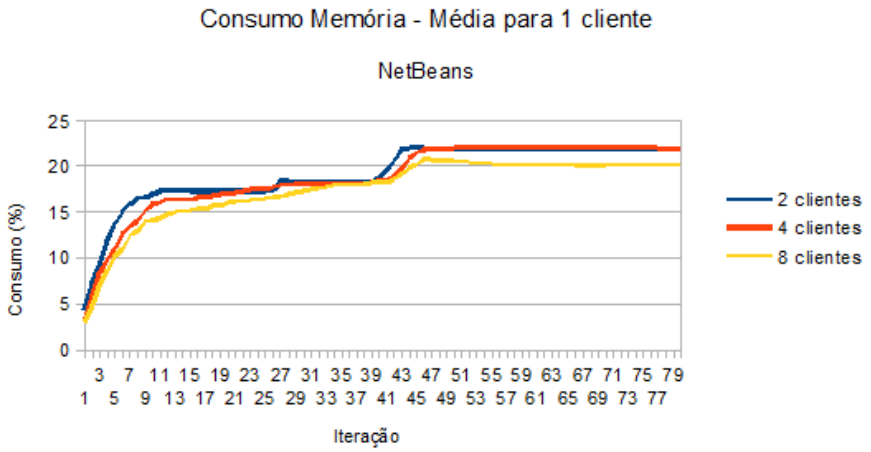


Figura 4.13: Consumo médio de memória para um cliente executando NetBeans

Diferente do que ocorreu com o Firefox, ao adicionar mais um servidor na arquitetura onde foi executado o NetBeans, esta aplicação utilizou maior capacidade de processamento e memória no total. Isso se deve ao fato de que essa aplicação já havia necessitado de mais capacidade de processamento e memória quando executada em apenas um servidor de aplicação, atingindo os limites do mesmo.

Podemos observar na Figura 4.14 que quatro clientes chegaram a consumir quase 130% da capacidade total de um dos dois servidores, totalizando um consumo de 260% da estrutura total, bem maior que aquilo que foi consumido quando existia apenas um servidor para responder às requisições de todos os clientes já que o máximo de capacidade de um servidor é apenas 200%.

Já nos testes com oito servidores, os servidores suportaram a carga gerada pelos clientes, diferente do que ocorreu com apenas um servidor. Porém, podemos observar que as aplicações executadas nos clientes exigiram algo em torno da capacidade total de três dos quatro processadores da estrutura.

Quando analisamos o gráfico da Figura 4.15, descobrimos que a aplicação não necessitava de tanta memória a mais do que foi a ela atribuída quando executada com um servidor. A execução com quatro clientes manteve praticamente a mesma porcentagem de utilização da memória, já que cada servidor gastou no máximo uma média de 40% da capacidade quando executados dois servidores e quando executado apenas um servidor este gastou cerca de 80% da capacidade. Já a execução com oito clientes utilizou

uma porcentagem maior de memória, praticamente o total da capacidade fornecida pelo servidor.

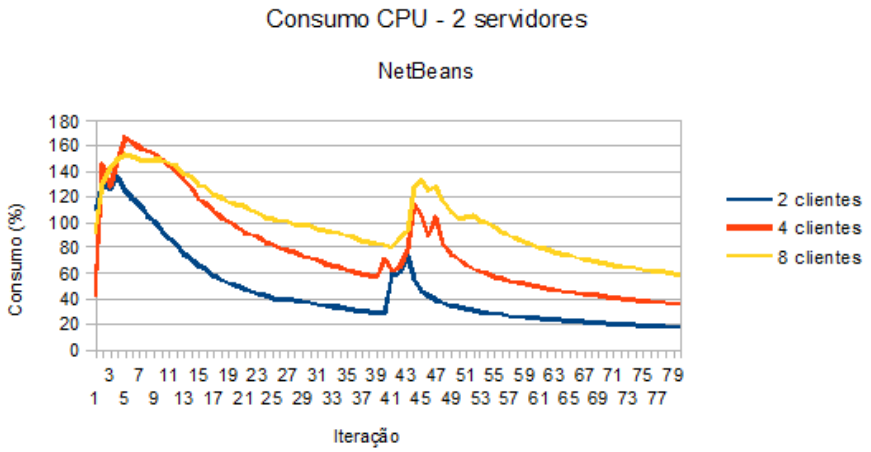


Figura 4.14: Consumo total de CPU executando NetBeans com dois servidores de aplicação

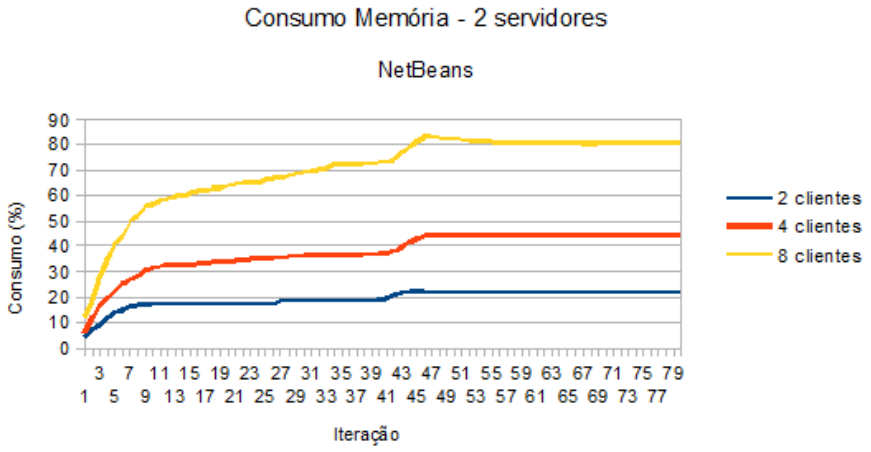


Figura 4.15: Consumo total de memória executando NetBeans com dois servidores de aplicação

4.3 Análise dos resultados

A partir dos resultados podemos concluir que, assim como foi dito na análise teórica, a estrutura LTSP necessita de uma rede rápida para a execução de suas tarefas. Isso ocorre porque todo o processamento dos dados dos clientes é feito através da mesma.

Porém, embora necessite de uma rede boa, a mesma não é um grande impedimento para o funcionamento da tecnologia LTSP. Essa conclusão ocorre do fato de que os maiores tráfegos gerados pelos testes em cenário crítico foram de 140 Mbps no *rootserver* e 280 Mbps no servidor de aplicação, este tráfego é aceito sem grandes problemas em uma rede de velocidade 100 Mbps.

O número de clientes é de suma importância neste cenário, nos resultados dos testes podemos constatar que a inicialização de dois clientes demorou um número bem menor de iterações para se concluir do que a inicialização de oito clientes.

Podemos estabelecer uma relação entre o aumento do tráfego nas interfaces de rede e o aumento do número de clientes na estrutura. Ao compararmos a duração da mais longa elevação de tráfego para quatro e para oito clientes notamos que, ao executarmos quatro clientes esta elevação durou cerca de 14 iterações e ao executarmos oito clientes esta elevação durou 32 iterações, um aumento de 129 % do número total de iterações. Esta elevação mais duradoura do tráfego com quatro clientes teve uma porcentagem parecida de aumento do número de iterações quando comparada com a duração da elevação mais duradoura na execução de dois clientes.

Isso nos mostra que à medida que dobramos o número de clientes, a carga de tráfego nas interfaces de rede da estrutura é multiplicada por aproximadamente 2,3 vezes o valor da carga de tráfego atual.

O LTSP responde bem quando é executado em seus clientes o Firefox. Mesmo com apenas um servidor de aplicação e oito clientes sendo executados, o servidor ficou a grande maioria do tempo com a metade de sua capacidade ociosa. A partir dos testes realizados com um único servidor de aplicação respondendo as solicitações dos clientes, podemos observar algumas características importantes do consumo do sistema.

Analisamos que, a medida que dobra o número de clientes, a capacidade de CPU exigida por esses clientes ao servidor aumenta em torno de 10%.

No aspecto da memória também é possível estabelecer uma relação entre o número de clientes e o consumo do servidor. Ao serem executados dois clientes o servidor teve até 15% de sua capacidade de memória consumida, ao serem executados quatro clientes, o valor de memória consumida por esses clientes dobrou, passando a ocupar aproxima-

damente 30% da capacidade total do servidor. O mesmo ocorre quando são atribuídos oito clientes ao servidor, a memória consumida atinge pouco mais de 60% da memória disponível aos clientes, o dobro do consumido por quatro clientes.

Os testes com Firefox nos permitem concluir, a partir desses resultados obtidos, que o consumo de memória do servidor tende a dobrar à medida que dobra o número de clientes, sendo que com dois clientes o consumo fica em torno de 15%.

Os testes realizados com o NetBeans nos mostraram que esta aplicação já exige uma capacidade bem maior dos servidores. Primeiramente, pelo fato de que com apenas um servidor respondendo solicitações de oito clientes, este servidor não suportou a carga de processamento exigida pelos clientes e ficou inoperante já no inicio da execução do teste. Isso indica que um único servidor, com as configurações já mencionadas neste trabalho, não suporta este número de clientes executando o NetBeans.

Quando permitimos a utilização de dois servidores, a estrutura responde consideravelmente bem a execução do NetBeans, é a partir dos testes com dois servidores que podemos analisar melhor alguns aspectos da estrutura LTSP na execução do NetBeans.

Podemos observar que na maior parte do tempo, naqueles momentos em que o consumo de cpu dos servidores é menos variável, à medida que dobramos o número de clientes o consumo de CPU em cada um dos servidores aumenta em torno de 20%, esses aumentos se dão a partir do consumo de CPU de cada servidor com a execução de dois clientes, que também é de 20%. Os maiores picos de consumo exibidos nos resultados dos testes, com oito clientes, atingem aproximadamente 140% da capacidade do servidor.

No aspecto de memória também podemos estabelecer uma relação entre o número de clientes e o consumo dos mesmos. Quando executados dois clientes, estes chegaram a requisitar algo em torno de 22% da capacidade de memória de cada servidor. Já quando foram executados quatro clientes, o consumo máximo gerou em torno de 43%, quase que o dobro do consumido pelos dois clientes. Ao executar oito clientes, o gráfico nos mostra que o consumo de memória de cada servidor foi aproximadamente o dobro do consumido por quatro clientes, atingindo 80%.

Isso nos mostra que, a partir do consumo de 22% na execução de dois clientes, o consumo de memória em cada um dos servidores tende a dobrar à medida que é dobrada

a quantidade de clientes do sistema.

5 Conclusão

Com base neste estudo, podemos concluir que a tecnologia LTSP é uma importante ferramenta para a criação de laboratórios de baixo custo e, principalmente, para o reaproveitamento de máquinas obsoletas.

Também podemos concluir que as exigências de investimento em servidores e em rede não inviabilizam a implementação de um projeto LTSP, principalmente naqueles casos em que os clientes executam aplicações mais leves. Isso nos é mostrado pela capacidade exigida pelos clientes na realização dos testes deste trabalho.

Outra contribuição desse trabalho foi o fornecimento de parâmetros de aumento de consumo a medida que é aumentada a quantidade de clientes na estrutura.

São possibilidades de trabalhos futuros a análise do trafego de rede na interface dos servidores no momento da execução de aplicações nos clientes, com estes já iniciados. Já que isso não foi realizado neste.

Também são possibilidades futuras de trabalho a análise do LTSP no que diz respeito à segurança dessa ferramenta e das tecnologias utilizadas por ela, além da realização de um estudo comparativo entre esta ferramenta e outras semelhantes como o *Thin Client Operating System* (TCOS) (Ortiz et al, 2012) e o *OpenThinClient* (Hedegren, 2011).

Outra possibilidade é a realização de testes de desempenho em laboratórios com maior quantidade de clientes e com clientes com capacidades individuais mais limitadas, para que isso permita confirmar na prática as conclusões obtidas neste trabalho.

Este trabalho espera ter contribuído com os seus objetivos, para o maior conhecimento e divulgação da tecnologia LTSP e desse modelo de processamento, além de ter contribuído para o auxilio nas implementações dessa solução.

Referências Bibliográficas

Achesinski, J.; Burokas, G. **Method and system for caching read requests to a shared image in a computer network**. In: Registro de Patente WO 2006047180 A1. PCT/US2005/037600, 2006.

Babu, B. R.; Parande, A. K. ; Basha, C. A. **A global environmental problem**. In: Electrical and electronic waste. Central Electrochemical Research Institute, Karaikudil, 2007.

Barbosa, A. A.; Barreto, L. P. **Avaliação de sistemas de arquivos distribuidos num ambiente de pequena escala**. In: Anais do XXVII Congresso da SBC. Universidade Federal da Bahia, 2007.

Bezerra, R. M. **A camada de transporte**. In: Redes de Computadores II. Saber Acadêmico, 2007.

Boito, F. **Estrategias para avaliação do desempenho do sistema de arquivos lustre**. In: Trabalho de Conclusão de Curso. Universidade Federal do Rio Grande do Sul, 2009.

Bugallo, A. M. D.; Barros, M. A. ; Torres, W. R. **Introdução ao dhcp**. In: Boletim bimestral sobre tecnologia de redes. Rede Nacional de Ensino e Pesquisa RNP, 1999.

Chao, T. **X display manager**. In: Linux XDMCP HOWTO. Linux Documentation Project, 2007.

Covelo, R. P. F.; Venda, P. J. L. **Segurança no acesso a sistemas embebidos**. In: licenciatura em Engenharia Eletrotecnica e de Computadores. Universidade Tecnica de Lisboa, 2005.

Ferretti, C. **Implementação de uma rede de computadores baseada no linux terminal server project**. In: Trabalho de Conclusão de Curso. Universidade federal de Lavras, 2004.

Gordon, B. R. **A dynamic model of consumer replacement cycles in the pc processor industry**. In: Marketing Science. Columbia Business School, 2009.

Hennessy, J. L.; Patterson, D. A. **A evolução da informatica e a tarefa do projetista de computadores**. In: Arqiutetura de Computadores, p. 3–7. Campus, 2003.

Hankins, D. **Dynamic host configuration protocol options used by pxelinux**. In: Network Working Group. Internet Systems Consortium, 2007.

Hedeggren, D. **Thin clients an open source product comparison**. In: Institutionen for kommunikation och information. University of Skovde, School of Humanities and Informatics, 2011.

Luis, A. F. B. M. F. **Portal de informação dum serviço de email**. In: Mestrado em Engenharia Informatica. Universidade de Lisboa, 2008.

Marimoto, C. E. **Configurando um servidor xdmcp**. In: Redes Guia Pratico. GDH Press e Sul Editores, 2006.

Marques, C. **Dhcp visão gerall**. In: Dynamic Host Configuration Protocol. clebermarques.com, 2007.

Michelin, A.; Talau, M. **Implantação de um sistema ltsp com debian gnu linux na utfpr campus dois vizinhos**. In: I Seminario de Produção Agropecuaria. Universidade Tecnologica Federal do Parana, 2007.

Morimoto, C. **Terminal burro**. In: Terminal Burro. Guia do Hardware, 2005.

Mooreira, A. **Especificação pxe**. In: Network Remote Boot. Instituto Pilitecnico do Porto, 2008.

Nieh, J.; Yang, S. J. ; NaomiNovik. **A comparison of thinclient computing architectures**. In: Network Computing Laboratory. Columbia University, 2000.

Ortiz, T.; Agustin, V. **Estudio comparativo de tecnologias de clientes ligeros ltsp tcos microsoft terminal server orientado a la reutilizacion de pcs**. In: Caso Practico Laboratorio de Computo de la Escuela Ruffo Didonato. Escuela Superior Politecnica de Chimmborazo, 2012.

Pereira, F. F. **Inicialização remota do thinstation para ambiente windows terminal service**. In: Curso de Especialização em Redes e Segurança de Sistemas. Pontificia Universidade Catolica do Parana, 2009.

Pires, A. S. **Implementação de um projeto piloto baseado em clientes magros na serpb**. In: Relatorio Final de Estagio. Instituto federal de Educaão, Ciência e Tecnologia da Paraiba, 2009.

Quillan, J. **Instalação dos utilitarios ltsp**. In: LTSP Linux Terminal Server Project v4.1. IEEE, 2006.

Ramos, C. C. L. C. **Vulnerabilidades e uso do ltsp**. In: Encontro Anual do IECOM em ComunicaçÃ$\mu es, Rede se Criptografia. Faculdade Mauricio de Nassau$, 2012.

Rodrigues, J.; Filitto, D. **Protocolos utilizados**. In: A Relevancia da Tecnologia LTSP na Inclusão Digital. Saber Acadêmico, 2007.

Ruppert, G. C. S. **Nfs guard uma solução de segurança para o protocolo nfs**. In: Dissertação de Mestrado. Fundação de Amparo á Pesquisa do Estado de SP, 2006.

dos Santos, R. H. S.; Braun, L. L. **Performance tests with ltsp**. In: Performance Tests with LTSP. IEEE Latin America Transactions, 2012.

da Silva, J. S.; Moraes, F. G. **Infraestrutura para controle de projetos em fpgas atraves do protocolo ethernet**. In: Infraestrutura para Controle de Projetos em FPGAS Atraves do Protocolo Ethernet. Faculdade de Engenharia PUCRS, 2011.

Sollins, K. **The tftp protocoll (revision 2)**. In: Network Working Group. MIT, 1992.

Sorj, B.; Guedes, L. E. **Problemas conceituais, evidências empiricas e politicas publicas**. In: Exclusão Digital. Novos Estudos, 2005.

Souza, D. A. P. F.; Ribeiro, E. M. ; Vieira, S. A. **Terminais leves**. In: Linux Terminal Server Project Como Instrumento de Comunicação Interna Para a FATEC. FATEC Tatui, 20012.

Tanenbaum, A. **Historia dos sistemas operacionais**. In: Sistemas Operacionais Modernos, p. 6–9. Pearson, 2008.

Timoteo, C. **Melhoria no processo de clonagem de computadores numa linha de produção**. In: Trabalho de Conclusão de Curso. Universidade de Pernambuco, 2005.

Tsai, C.; Lee, T. **A remote collaboration system design and construction**. In: ACM. Association for Computing Marchinery, 2011.

Vieira, W. V. **Fundamentação teorica**. In: Mobilidade com o Protocolo SIP Concepção de uma Plataforma de Testes e Analise de Cenarios. Instituto Federal de Santa Catarina, 2010.