UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

CRISTIANO OENNING

AVALIAÇÃO DE DESEMPENHO E OTIMIZAÇÃO DE SERVIDORES WEB APACHE

CRICIÚMA, JULHO DE 2007

CRISTIANO OENNING

AVALIAÇÃO DE DESEMPENHO E OTIMIZAÇÃO DE SERVIDORES WEB APACHE

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação na Universidade do Extremo Sul Catarinense.

Orientador: Prof. M. Sc. Paulo João Martins

CRICIÚMA, JULHO DE 2007

Cristiano Oenning

Avaliação de Desempenho e Otimização de Servidores Web Apache

Submetido ao corpo docente do Departamento de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

_____

Profa. M. Sc. Ana Cláudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

_____

Prof. M. Sc. Paulo João Martins
Orientador

_____

Prof. M. Sc. Rogério Antônio Casagrande

_____

Prof. Esp. Arildo Antônio Sônego

Dedico a meus pais, que nunca mediram esforços para me ajudar a alcançar meus objetivos, sempre deixando livre para que eu mesmo escolhesse o caminho a seguir, contudo, tentando sempre mostrar a diferença entre o que é certo e errado.

AGRADECIMENTOS

A Deus, por ter me proporcionado mais esta conquista em minha vida.

Aos meus pais, Valmiro e Bernadete pela compreensão e por todo o apoio que recebi durante toda a minha vida, pela importante contribuição em minha formação de caráter e intelectual, valores que continuarão para sempre comigo.

Á minha mulher, Fernanda e meu filho André, que me dão força para continuar sempre, apesar das diversas dificuldades que encontramos em nosso caminho.

Aos meus amigos que sempre me ajudaram.

Ao professor Paulo João Martins pelo conhecimento e orientação durante a realização deste trabalho.

A todos os professores que de alguma forma contribuíram para o meu aprendizado e formação durante a realização deste curso.

“O pessimista queixa-se do vento, o otimista espera que ele mude e o realista ajusta as velas.“

(Willian George Ward)

RESUMO

Devido à explosão tecnológica dos últimos tempos e a expansão da Internet que através, principalmente, da Web vem tornando-se uma ferramenta indispensável para todas as áreas de atuação, fica claro a necessidade de que este tipo de serviço seja dinâmico e eficiente. Este trabalho propõe um estudo sobre este assunto, mais especificamente na área de servidores com o intuito de se conseguir o máximo desempenho para uma dada configuração de hardware, trabalhando as configurações de software. Realizou-se uma pesquisa, necessária para fornecer um embasamento sobre o tema e posteriormente testes práticos com as variáveis comprovando ou não sua influência no que diz respeito ao desempenho dos servidores e conseqüentemente dos serviços da Web. Utilizando-se de ferramentas de apoio nas tarefas de monitoramento de servidores e gerenciamento de rede, foram realizadas medições para latência e *throughput* das informações a cada alteração efetuada conseguindo determinar a influência e importância de se ajustar corretamente as configurações do sistema a cada condição específica de trabalho. O estudo concentrou-se mais no aspecto de melhorias no software servidor e conexões TCP podendo fornecer importantes informações a pessoas que tenham interesse nesta área de atuação.

Palavras-chave: servidores; Web; Internet; desempenho; otimização;

ABSTRACT

Due to the explosion of technology in the last years and the expansion of the Internet that, mainly through the Web, has been becoming a fundamental tool for all action areas, it is evident that this kind of service must be dynamic and efficient.

This paper proposes a study about this theme, more specifically in the server areas with the aim of getting the greatest performance of a given configuration of hardware, working the configurations of software.

A research was carried out, which was necessary to provide a solid foundation about the theme, and later, practical tests with the variables were also carried out to prove or not its influence regarding the performance of servers and consequently the Web services.

Measures for latency and throughput of information at each alteration performed managing to determine the influence and importance of adjusting the configurations of the system correctly at each specific condition of work were made by using helpful tools in the tasks of monitoring of servers and net managing the net.

The study mainly focused on the aspect of improvement of the server software and connections TCP, being able to provide important information for people who have interest in this action area.

Key words: server; Web; Internet; performance; optimization.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

ADSL	*Asymmetric Digital Subscriber Line*
API	*Application Programming Interface*
CPU	*Central Processing Unit*
CRLF	*Carriage Return Line Feed*
DHTML	*Dynamic HyperText Markup Language*
DNS	*Domain Name System*
FTP	*File Transfer Protocol*
HD	*Hard Disk*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IP	*Internet Protocol*
MIME	*Multipurpose Internet Mail Extensions*
MRTG	*Multi Router Traffic Grapher*
MSS	*Maximum Segment Size*
MTU	*Maximum Transmit Unit*
OSI	*Open Systems Interconnection*
PC	*Personal Computer*
RAM	*Random Access Memory*
RTT	*Round Trip Time*
RWIN	*Receive Window*
SGML	*Standard Generalized Markup Language*
SMTP	*Simple Mail Transfer Protocol*
SNMP	*Simple Network Management Protocol*

SSL	*Secure Socket Layer*
TCP	*Transmission Control Protocol*
TLI	*Transport Layer Interface*
TTL	*Time to Live*
UDP	*User Datagram Protocol*
URIs	*Uniform Resource Identifiers*
URL	*Uniform Resource Locator*
WAST	*Web Application Stress Tool*

SUMÁRIO

1 INTRODUÇÃO

Um dos pontos mais importantes para o sucesso de uma empresa é conseguir implantar uma estratégia de marketing adequada e que realmente traga resultados positivos para o negócio. Conforme Hirata (2002) uma das ferramentas mais utilizadas atualmente para isso é a exposição dos produtos pela Internet por meio de *sites*, pois muitas vezes é através do comércio eletrônico que se consegue ampliar as transações comerciais e aumentar o faturamento da empresa. Contudo, o crescente uso deste meio vem impondo uma carga bastante alta de trabalho à rede e aos servidores Web o que tende a tornar o serviço mais lento.

Para evitar que o processo fique lento e inviável é necessário que se faça uma avaliação do desempenho do sistema e da carga de trabalho imposta, e assim, se necessário, realizar melhorias que podem aumentar a capacidade dos recursos de hardware ou trabalhar nos softwares para que se tornem mais leves e dinâmicos.

Estas melhorias necessárias certamente podem trazer benefícios, contudo, representam custos para as empresas, por isso é importante que se consiga o melhor aproveitamento possível dos recursos disponíveis, daí a importância da otimização de desempenho do sistema.

1.1 OBJETIVO GERAL

Demonstrar algumas formas práticas para a avaliação e melhoria do desempenho de servidores Web, utilizando estudos de caso.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos são:

a) compreender formas de medidas de desempenho para avaliação de servidores Web, onde se consiga medir a carga imposta (métricas de desempenho);
b) verificar a carga de trabalho a que está sendo submetido o servidor;
c) propor estudo de casos com a finalidade de obter um melhor desempenho no tempo de resposta do servidor Web;
d) avaliar o ganho de desempenho obtido pelo servidor;
e) comparar os resultados obtidos, de forma a comprovar ou não a melhoria de desempenho.

1.3 JUSTIFICATIVA

A expansão da Internet vem ocorrendo de forma muito acelerada, tornando-se cada vez mais acessível a todos, sem impor restrições quanto à plataforma ou qualquer tipo de software utilizado. Para isso, os desenvolvedores trabalham para aperfeiçoar seus programas deixando-os cada vez mais complexos e robustos e assim, por conseqüência, maiores e mais pesados.

A crescente quantidade de usuários e de informações disponíveis aumentam o tráfego, congestionando as redes utilizadas e sobrecarregando os servidores Web, tornando-os mais lentos. O desempenho é uma das características mais visíveis de qualquer sistema, e ele freqüentemente ocupa um lugar de destaque na lista de reclamações dos usuários, por isso é muito importante que seja dada uma atenção especial a este fato (RESELLER, 2007).

Conforme Hirata (2002) a resposta mais fácil para a resolução do problema de desempenho poderia ser melhorias de hardware, contudo estas melhorias implicam em custos elevados. Para que se consiga otimizar a utilização dos recursos de hardware é necessário que se faça uma avaliação correta de desempenho, avaliando e reconfigurando os programas manipulados, utilizando o que realmente é necessário e da forma correta, melhorando assim a relação custo/benefício do investimento.

1.4 ESTRUTURA DO TRABALHO

O desenvolvimento deste projeto trata da avaliação e possíveis otimizações relacionadas a servidores Web, tema este que aparece hoje com muita freqüência em pesquisas e trabalhos correlatos, devido à importância que o assunto vem adquirindo, porém, quase sempre feito de uma forma bastante pontual, destinando-se a resolver problemas específicos e não buscando abordar o assunto de forma mais ampla.

Este trabalho está descrito em 6 capítulos.

O Capítulo 1 trata da introdução, onde se buscou demonstrar qual o problema em questão e quais os objetivos a serem alcançados com a realização do trabalho.

O Capítulo 2 apresenta um conceito de Internet, que serve para formar um embasamento ao assunto do trabalho.

No Capítulo 3 é descrito sobre o conjunto de protocolos TCP/IP, assunto indispensável para a compreensão do funcionamento da comunicação via rede, bem como a estrutura de protocolos envolvida e possíveis influências no desempenho.

No Capítulo 4 é apresentado um conceito para Web, seus componentes semânticos principais e suas funções. Trata também do conceito de Servidor, seu funcionamento, arquiteturas e do software servidor Apache.

O Capítulo 5 fala sobre avaliação de desempenho propriamente dita, parâmetros em questão, conceitos relacionados, medição e avaliação de tráfego, bem como as técnicas para monitoração de tráfego na Web.

No Capítulo 6 é apresentado o trabalho realizado em si, seguindo como proposto no projeto, realizando estudos sobre o assunto e implementando de forma prática.

Por fim tem-se a conclusão com as considerações finais sobre as atividades realizadas e sugestões para trabalhos futuros relacionados com o tema estudado.

2 A INTERNET

Para Kurose e Ross (2006) devido a sua complexidade, fica bastante difícil para se definir a Internet com apenas uma frase. Seus componentes, tanto hardwares e também softwares mudam e evoluem constantemente quanto aos seus serviços. Por este motivo, talvez seja mais correto fazer uma abordagem descritiva e detalhada de seus componentes de hardware e software.

O termo Rede de Computadores já está um pouco desatualizado, isto porque já não são mais só computadores ligados à Internet, mas sim vários tipos diferentes de equipamentos que podem ser denominados de hospedeiros ou sistemas finais. Estes, por sua vez, são conectados por meio de diferentes tipos de meios físicos, podendo ser cabos, fibra óptica ou até ondas de rádio. A informação a ser transmitida de um sistema final (emitente) para um receptor é feita utilizando uma técnica chamada de comutação de pacotes. Os comutadores (*switches*[1] e roteadores[2]) fazem a organização das informações (pacotes) na rede apontando a rota desde o emitente até o receptor. Os pacotes percorrem uma seqüência de enlaces e comutadores por meio da rede partindo do sistema final remetente até o sistema final receptor.

Os sistemas finais acessam a Internet por meio de provedores de serviço, estes por sua vez se comunicam com outros provedores, que podem ser de um nível mais alto ou mais baixo e assim permitem que os usuários tenham acesso a toda a rede.

Para que esta comunicação seja possível e eficiente, são usados protocolos de comunicação que fazem o controle de transmissão das informações. Os principais protocolos utilizados para comunicação na Internet são conhecidos como TCP/IP, onde TCP significa *Transmission Control Protocol* que é responsável pelo controle de

[1] Equipamento que cria um canal de comunicação exclusiva entre a origem e o destino para que os dados sejam repassados.

transmissão e IP significa *Internet Protocol* que especifica o formato dos pacotes (KUROSE; ROSS, 2006).

Existem ainda outros protocolos importantes como o HTTP que também será descrito no decorrer deste trabalho.

[2] Equipamento que tem a mesma função de um switch, contudo, tem a capacidade de escolher a melhor rota para os pacotes de dados entre origem e destino.

3 TCP/IP

Surgiu originalmente da ARPANET e tornou-se com o passar do tempo um protocolo universal das redes de computadores. Como já mencionado anteriormente, o TCP/IP fornece a idéia de apenas um protocolo, quando na realidade compreende um conjunto de protocolos onde os dois principais são o TCP e o IP (LIMA, 2003).

3.1 FUNÇÕES DO TCP/IP

As funções básicas deste conjunto de protocolos são:

a) dividir as informações (ou mensagens) a serem transmitidas em segmentos gerenciáveis;
b) fazer a interface com o hardware adaptador da rede;
c) endereçamento: O remetente deve ser capaz de endereçar corretamente os dados ao receptor e este, por sua vez, deve ser capaz de reconhecê-lo;
d) roteamento: fazer o roteamento correto das informações mesmo que a origem e o destino estejam localizados em sub-redes ou em redes distintas;
e) garantir o controle de fluxo, a verificação de erros e confirmações de recebimento;
f) passar os dados das aplicações para a rede;

Este trabalho não tem como objetivo fazer um estudo aprofundado ou se ater a conceitos de redes de computadores ou redes de comunicação quaisquer, porém, são necessários alguns conceitos básicos no assunto para apresentar embasamento ao tema estudado. Para isso será feita uma breve descrição do conceito de redes e do modelo de referência (modelo OSI) que serve como um padrão para desenvolvimentos nesta área.

3.2 REDE

Coleção de computadores ou dispositivos semelhantes que podem comunicar-se por um meio de transmissão comum (GALLO; HANCOCK, 2002).

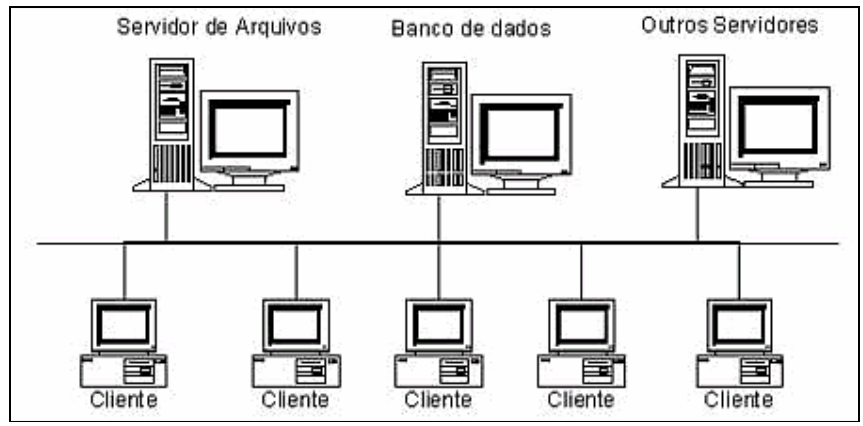

Figura 1. Exemplo de rede de Computadores
Fonte: BATTSTI, J. (2003)

Computadores interligados em uma rede devem ser capazes de trocar mensagens e informação e entendê-las. Assim, como os computadores rodam aplicações que gerenciam entradas e saídas, é necessário que algumas dessas aplicações também sejam capazes de se comunicar com aplicações de outros pontos da rede. Ao conjunto de regras que definem uma padronização para esse processo de comunicação dá-se o nome de protocolo de rede.

3.3 ENCAPSULAMENTO

Quando uma aplicação envia dados usando TCP/IP, ela os envia por meio de cada nível da pilha de protocolos. Cada camada possui informações relevantes e que deverão ser anexadas aos dados oriundos da camada superior em forma de cabeçalho.

Os dados mais o cabeçalho HTTP formam a mensagem, e quando empacotados com o cabeçalho TCP formam o segmento TCP, que empacotado no cabeçalho IP forma um datagrama e por fim quando adicionado o cabeçalho de Internet forma os frames que posteriormente formam um fluxo de *bits* para a rede. Então os dados vão sendo empacotados dentro de cada cabeçalho formando um pacote de dados em forma de pilha. No final, os dados são enviados como uma seqüência de *bits*, pela rede (CASAD; WILLSEY, 1999).

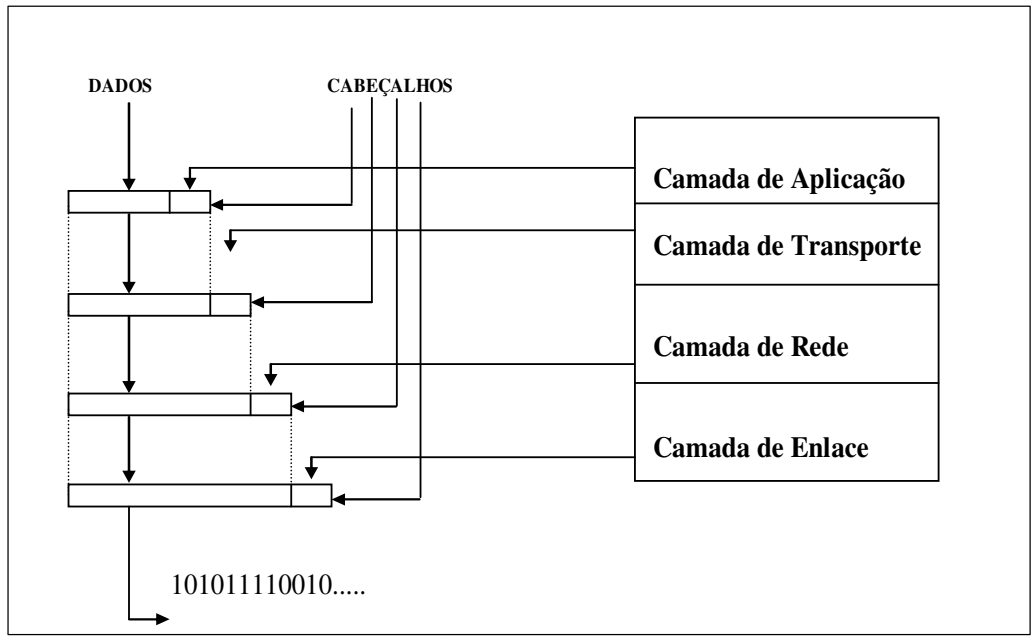

Figura 2. Ilustração da formação e envio dos pacotes de dados.
Fonte: Adaptado de CASAD, J.; WILLSEY, B.(1999).

No recebimento os dados são desempilhados na seqüência inversa e cada protocolo busca as informações em seus respectivos cabeçalhos, possibilitando assim a recomposição da mensagem original.

3.4 TCP

O TCP é o responsável por garantir que os dados enviados chegarão ao seu destino e, além disso, de que chegarão intactos. Pode se dizer que este seja um protocolo orientado à conexão, isto é, antes de os dados serem enviados, a máquina origem entra em contato com a máquina destino e estabelece regras para a transferência, criando assim, uma espécie de duto de dados pela rede (SANTOS, 2005).

Atuando na camada de transporte, provê serviços para a camada de aplicação. Conforme Hirata (2002) sua função principal, independentemente da rede ou do sistema utilizado, é garantir a conexão entre processos de máquinas distintas da rede, de forma confiável, com o fluxo de dados controlado e transmissão bidirecional, ou seja, os dois pontos da rede podem transmitir dados simultaneamente e em ambos os sentidos.

Os dados são enviados em segmentos TCP, que cruzam a Internet em pacotes IP. O conjunto completo do protocolo é freqüentemente chamado de TCP/IP porque como já mencionado anteriormente, são os dois protocolos fundamentais. Sendo assim, este protocolo merece atenção especial já que possui influência direta no desempenho da Web e dos seus servidores.

3.4.1 APIs - *Application Program Interface*

Pode ser definido como um conjunto normalizado de rotinas e chamadas de software que podem ser referenciadas por um programa aplicativo para acessar serviços essenciais de uma rede (COLOMBO, 2002). Essas interfaces de programação (APIs), agrupam funções que são as responsáveis pelos serviços de comunicação providos pela rede.

Estes serviços são usados pelo código presente no cliente para enviar suas solicitações de serviço e recebimento das respostas, e também pelo código presente no servidor para receber solicitações e enviar as respostas.

No caso mais específico da Internet, onde os protocolos utilizados são o TCP/IP, existem duas interfaces de programação, os *sockets* e *Transport Layer Interface* (TLIs), mas as mais populares são mesmo os *sockets*, os quais serão abordadas no decorrer deste trabalho.

3.4.1.1 *Sockets*

Para que dois processos de máquinas diferentes consigam trocar informações é necessária a utilização de *sockets*, forma através da qual a camada de transporte fornece suporte para serviços de comunicação à camada de aplicação. Este, por sua vez, é formado pelo IP mais a porta de comunicação, que é um número interno de 16 *bits* e que corresponde ao endereçamento interno à máquina (FERREIRA, 2006).

3.4.2 Controle de Congestionamento

Emissores de TCP controlam o congestionamento da rede diminuindo ou aumentando a taxa de transmissão, evitando assim a sobrecarga da rede e o aumento de perdas de pacotes. O emissor controla esse ritmo através de uma janela deslizante à qual depende do espaço disponível no *buffer*[3] do receptor e da largura de banda da rede.

Então, quando for detectada a perda de algum pacote o emissor diminuirá o tamanho dessa janela, caso contrário vai aumentando gradualmente, buscando a maior velocidade de transmissão sem perdas de informação.

O aumento dessa janela e, conseqüentemente, da taxa de transmissão acontece de forma aditiva e gradual, já a sua diminuição acontece de forma multiplicativa, ou seja, muito mais acentuada.

Uma nova conexão deve começar com um tamanho de janela pequeno para não congestionar inicialmente a rede, já que ainda não possui informação sobre o tráfego na rede. Nesta primeira etapa, a cada confirmação de recebimento do pacote, o emissor dobra o tamanho da janela até o momento em que começa a haver perda de pacotes. Então, bruscamente o tamanho da janela é reduzido à metade do tamanho anterior ao congestionamento, voltando em seguida a ser incrementada, só que agora unitariamente. Contudo, esta fase, conhecida como fase de partida lenta, provoca perdas em velocidade de transmissão, já que a largura de banda não é totalmente utilizada durante este intervalo de tempo (HIRATA, 2002).

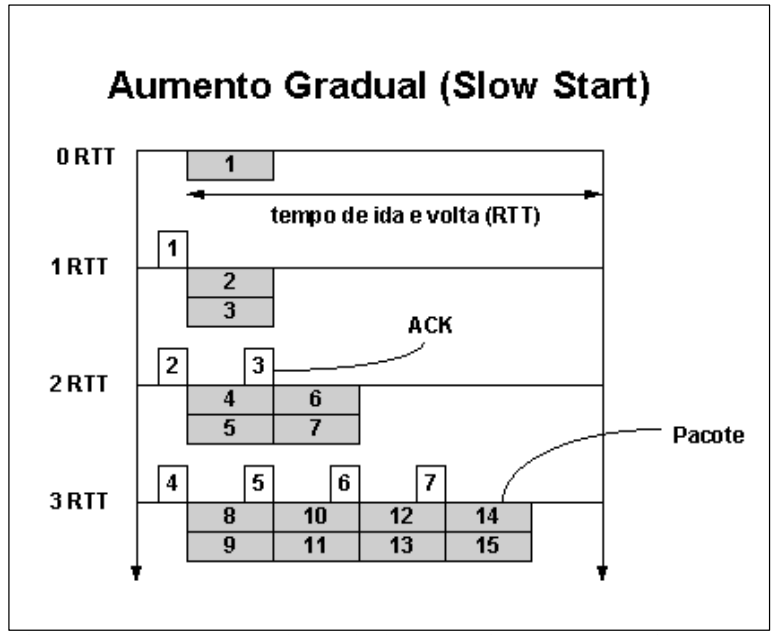

Figura 3. Comportamento da Janela de Congestionamento no acionamento.
Fonte: FILHO, N.; DIAS, A.; CRUZ, L. (1999)

[3] Região de memória temporária utilizada para leitura e escrita de dados.

Esta ainda é uma área bastante pesquisada, pois um controle de congestionamento conservador provoca perdas significativas no desempenho da transmissão de pacotes e o não controle provocaria problemas de estouro de *buffer*, congestionamento da rede e, conseqüentemente, perdas de pacotes e retransmissão de dados. O mesmo vale dizer para valores de tempo limite para retransmissão de pacotes, já que um tempo muito baixo pode gerar uma retransmissão desnecessária e o contrário pode atrasar a detecção de perdas de pacotes de dados. O responsável por verificar a perda de pacotes é o emissor do TCP, e isso pode ser detectado de duas maneiras: o tempo limite para retransmissão excedido ou uma confirmação duplicada (KRISHNAMURTHY; REXFORD, 2001).

3.5 IP

Cada host de uma determinada rede possui uma identificação que possibilita o encaminhamento (roteamento) dos pacotes de dados pela rede. Este número de endereçamento, denominado IP, é formado por uma combinação de 4 bytes (ou 32 bits), sendo que cada um pode assumir valores de 0 a 255 (SOUZA, 2001).

Exemplo: 192.105.003.11

A figura 6 demonstra a formatação geral dos pacotes IP.

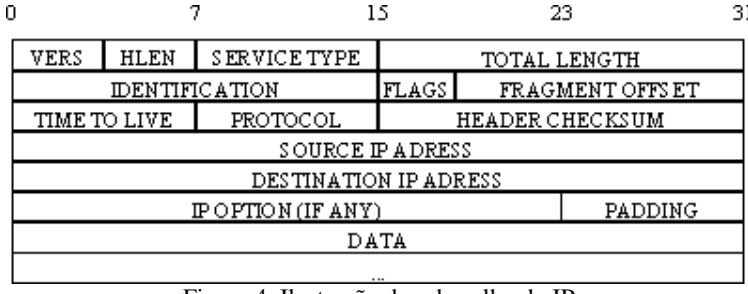

Figura 4. Ilustração de cabeçalho do IP
Fonte: Adaptado de JUNIOR, R. (2007)

Veja no quadro abaixo a descrição dos campos utilizados em um cabeçalho IP:

Campo	Descrição
VERS	Trata-se da versão do IP utilizada
HLEN	Tamanho do cabeçalho do pacote IP
SERVICE TYPE	Indica às sub-redes o tipo de serviço que deve ser oferecido ao Datagrama
TOTAL LENGTH	Tamanho total do pacote IP
IDENTIFICATION	Número de identificação do Datagrama
FLAGS E FRAGMENT OFFSET	Indica se a mensagem enviada foi ou não fragmentada
TIME TO LIVE	É o tempo que o pacote tem para encontrar o seu destino na rede, caso contrário, é descartado
PROTOCOL	Indica se o protocolo de nível superior é o TCP ou UDP
HEADER CHECKSUM	Controla os erros de cabeçalho do pacote IP
SOURCE IP ADDRESS	Identifica o endereço IP de origem
DESTINATION IP ADDRESS	Identifica o endereço IP de destino
IP OPTION	Especifica o tipo de pacote IP (dados ou controle)
PADDING	Este campo possui tamanho variável e é utilizado para se garantir que o comprimento do cabeçalho do datagrama seja sempre um múltiplo inteiro de 32 bits
DATA	Dados transportados

4 A WEB

A Web corresponde a todo o universo de informações disponíveis na rede. Utilizando-se uma interface gráfica e intuitiva, é possível através de *hiperlinks* acessar páginas e conteúdos sem preocupação com formatos ou locais de origem. Pode ser definida como uma aplicação em rede que por meio de computadores liga usuários e serviços do mundo inteiro (KRISHNAMURTHY; REXFORD, 2001).

É formada por três componentes semânticos principais; as URIs, que são mecanismos de nomeação universais para identificar recursos na Web, o HTML que é uma linguagem para criação de hipertexto e o HTTP que é o protocolo utilizado para comunicação entre cliente e servidor.

4.1 URI - *Uniform Resource Identifier*

Os acessos e a manipulação de recursos na Web são realizados através do URI, que serve para identificação destas solicitações. Seu formato mais comum é o *Uniform Resource Locator* (URL) que consiste no endereço utilizado pelo navegador para as páginas.

Pode-se dizer que um URI é formado por três partes: o protocolo de comunicação, o nome do servidor e o recurso desejado (KRISHNAMURTHY; REXFORD, 2001).

Os servidores DNS são os responsáveis pela conversão das URLs em endereços IPs ou vice-versa.

4.2 HTML - *HyperText Markup Language*

Derivada da *Standard Generalized Markup Language* (SGML), o HTML não é uma linguagem de programação, mas um padrão de estruturação e representação de documentos de hipertexto. A linguagem permite, além da estruturação de textos, a referência a outros documentos ou até mesmo às imagens.

Apesar de suas excelentes qualidades, o HTML é uma linguagem bastante limitada no que diz respeito à interação com o usuário, trabalhando apenas com conteúdo estático e sem possibilitar o envio ou a busca de dados. Para que isso se torne possível então, são utilizadas outras linguagens de programação como o PHP ou o Java que desempenham esse papel de manipulação de dados. A essa junção do HTML com outras linguagens de programação dá-se o nome de *Dynamic HTML* (DHTML) (FOUCES, 2004).

4.3 HTTP - *HyperText Transfer Protocol*

O HTTP, que atua na camada de aplicação, é o protocolo central da Web, padrão de transferência de conteúdo nos servidores. Nessa estrutura cliente/servidor, no lado cliente estão os *browsers* (ou navegadores) como o Internet Explorer e o Netscape, já no lado servidor, estão os softwares servidores como o Apache ou o IIS da Microsoft. É o responsável por definir a estrutura das mensagens e a forma como essa comunicação entre cliente e servidor é realizada (HIRATA, 2002).

Até 1997 utilizou-se a versão 1.0 do HTTP em *browsers* e servidores, a partir daí então foi disponibilizada uma nova versão, a versão 1.1 do protocolo que devido a mudanças realizadas trouxe bastante ganho no que diz respeito a desempenho, passando assim, gradativamente, a substituir a versão anterior. A migração de uma versão para outra não apresentou qualquer tipo de problema já que as duas versões são compatíveis.

4.3.1 Funcionamento Básico

No processo cliente/servidor pode-se dizer que os servidores ficam "escutando" a porta 80, que é a porta do TCP por onde os clientes enviarão as suas solicitações. A partir daí o processo acontece da seguinte forma:

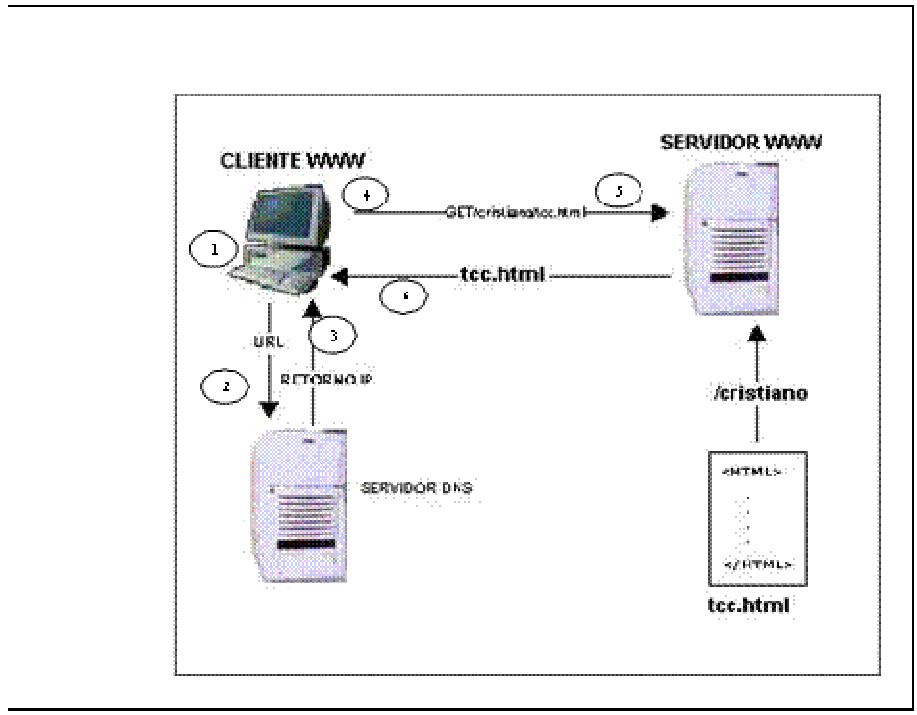

Figura 5. Esquema de funcionamento do servidor
Fonte: Adaptado de LIMA, João P. (2003).

1 – O cliente (usuário) digita no navegador o endereço da página, ou melhor, informa a URL da página na qual deseja buscar algum recurso que esteja disponível.

2 – O navegador informa ao Servidor DNS o endereço digitado pelo usuário e aguarda uma resposta.

3 – O Servidor DNS então verifica qual é o endereço IP, ou seja, em que local da rede está o recurso que está sendo solicitado e retorna para a aplicação.

4 – Com este endereço IP o navegador estabelece uma conexão TCP com o servidor, que recebe um sinal na porta 80 à qual fica monitorando constantemente.

5 – Em seguida, através do comando do HTTP, neste caso será exemplificado utilizando o comando GET que é o mais comum, o navegador solicita o recurso desejado ao servidor.

EX.: GET/cristiano/tcc.html

6 – Por fim o servidor envia o arquivo solicitado, que nesse caso seria **tcc.html** e o navegador o apresenta já formatado ao usuário.

O funcionamento do protocolo HTTP se baseia em solicitações ao servidor, cujas conexões são temporárias, ou seja, assim que a transferência do conteúdo foi efetuada o servidor desconecta este cliente (LIMA, 2003).

4.3.2 Requisições

Uma requisição feita a um servidor é formada basicamente por duas partes que são separadas em linhas de comando. A primeira linha compreende a Linha de Requisição e as outras linhas subseqüentes são as Linhas de Cabeçalho (FILHO; BIANCHINI, 2003).

4.3.2.1 Linha de Requisição

Formada basicamente por três partes que seriam dispostas na seguinte ordem:

1º - Método: Indicam o tipo de operação a ser realizada. São vários os métodos existentes, entre eles estão o método HEAD que retorna informações sobre o objeto como sua data de criação ou alteração e o tamanho do objeto, o método POST através do qual são enviadas as informações ao servidor e vários outros métodos como PUT, DELETE, OPTIONS, CONNECT, TRACE e o mais comum de todos eles que é o método GET o qual retorna o objeto (informação) solicitado pelo cliente.

2º - URL: Indica o objeto que está sendo solicitado pelo cliente.

3º - Versão: Indica a versão utilizada do protocolo HTTP.

4.3.2.2 Linhas de Cabeçalho

As linhas de cabeçalho (*header lines*) são responsáveis por informar o servidor sobre os parâmetros que o cliente está enviando, bem como informar ao cliente sobre a natureza da resposta enviada pelo servidor. Podem ser compostas por várias

opções, sendo cada uma delas separadas em linhas diferentes e dispostas no formato **Opção (header field name): Valor (value)** terminando com um CRLF (enter).

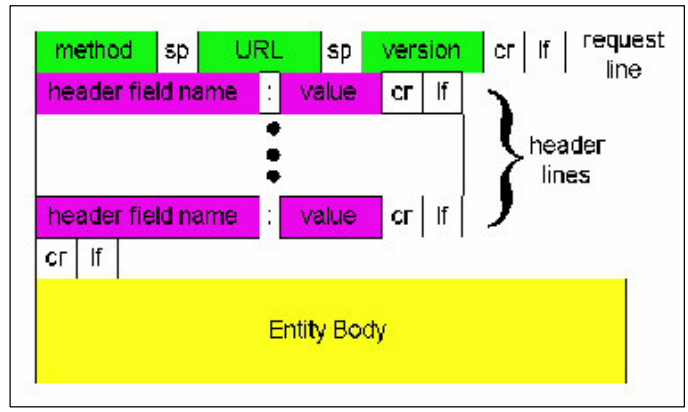

Figura 6.Mensagem de pedidos http: Formato Geral.
Fonte: MARTINS, J. (2007)

A opção Corpo da Entidade (*Entity Body*) não é usada no método GET, mas sim no método POST, ou seja, quando o cliente está enviando informações ao servidor.

4.3.3 Respostas

De uma forma geral, as mensagens de resposta possuem primeiramente uma linha de status, posteriormente o cabeçalho e por fim o corpo da mensagem. Estes são separados em linhas por CRLF (enter) (CANTU, 2003).

4.3.3.1 Linha de Status

A linha de Status é formada pela versão do http utilizada no servidor, o código de resposta ou código de status e mais a mensagem de resposta. Entre os códigos de status e mensagens mais comuns estão (FILHO; BIANCHINI, 2003):

• 200 OK – Indica o sucesso da requisição.

• 301 *Moved Permanently* – Indica que o recurso solicitado mudou de localização, e a mensagem mais adiante traz sua nova localização (*Location*:).

• 400 *Bad Request* – Servidor não reconheceu a mensagem de pedido.

• 404 *Not Found* – Recurso solicitado não se encontra neste servidor.

• 505 HTTP *Version Not Supported* – Versão HTTP imcompatível.

4.3.3.2 Cabeçalho

O formato do cabeçalho de uma resposta em um pedido http é o mesmo das requisições, sendo formado por opções e valores.

Entre as opções de destaque pode-se citar a opção *Date* que corresponde ao dia e hora da geração da resposta, a opção *Last-Modified* ou última modificação do objeto, a opção *Server* que identifica o servidor que gerou a resposta e a opção *Keep-Alive* que determina o tempo de persistência de uma conexão inativa.

Essas são apenas algumas das opções, mas existem muitas outras que auxiliam na formatação e identificação de requisições e respostas.

Como este trabalho está relacionado com o desempenho dos servidores, é indispensável ressaltar a importância da opção *Last-Modified* (última modificação). Caso o *browser* já possua em *cache*[4], o recurso em questão, o servidor não precisa transmitir o recurso, mas sim uma mensagem validando esta opção, ou seja, indicando que o recurso não sofreu mais alterações. Esta opção pode representar um ganho expressivo de desempenho já que o mesmo cliente pode requisitar mais de uma vez, ou inúmeras vezes, o mesmo recurso.

4.3.3.3 Corpo da Entidade

Partes da mensagem aonde vão os dados propriamente ditos. O recurso pode simplesmente ser anexado ao cabeçalho (por meio da URL), em alguns casos gerado através de *scripts*[5] executados no servidor, ou até mesmo através de buscas a bancos de dados.

A figura 10 representa o formato geral para mensagens de resposta a requisições http.

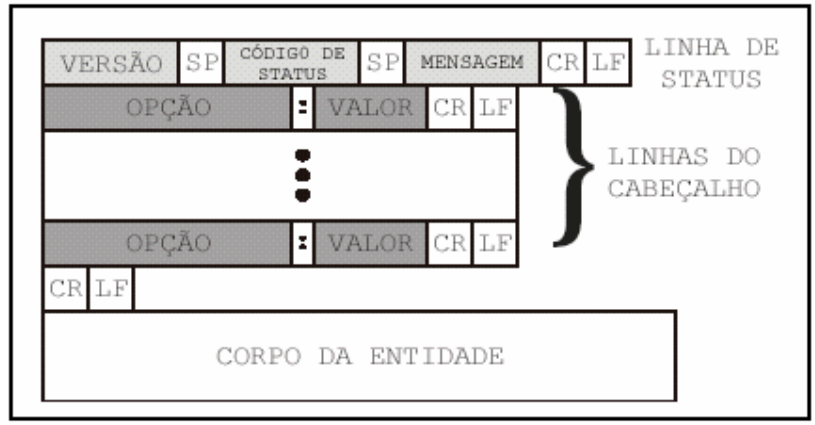

Figura 7.Formato geral de uma resposta a uma requisição HTTP.
Fonte: HIRATA, R. (2002).

4.4 SERVIDORES WEB

Para Krishnamurthy e Rexford (2001) um servidor Web é um programa que trata de pedidos http por recursos específicos feitos pelo cliente, gerando e transmitindo respostas. Entre estes o mais conhecido e utilizado é o Apache. Os servidores utilizam-se do protocolo HTTP e atuam na camada de aplicação da Internet conforme mostra a figura 11.

[4] Pequena quantidade de memória estática de alto desempenho, que tem por finalidade melhorar a performance do processador, fazendo uma busca antecipada na memória RAM.
[5] Arquivos texto com um conjunto de instruções que dizem o que o programa deverá fazer.

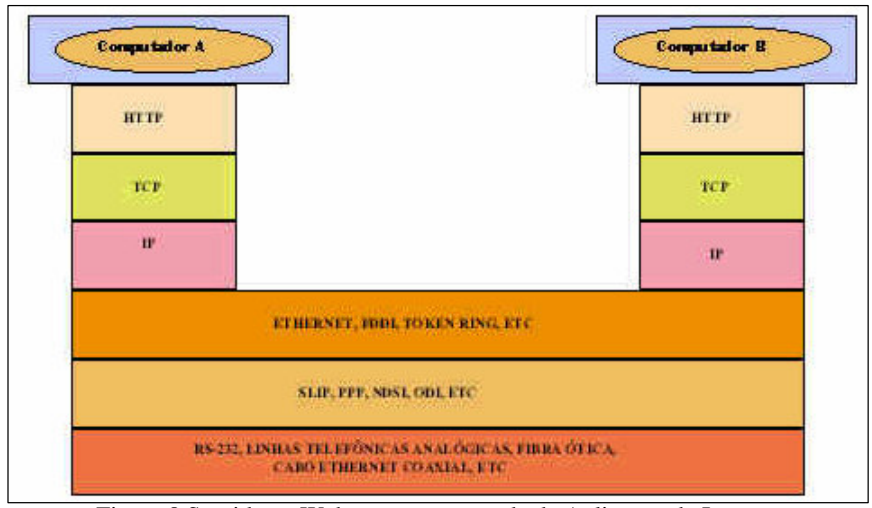

Figura 8.Servidores Web atuam na camada de Aplicação da Internet
Fonte: ZOTTO, O. (2006).

4.4.1 Funcionamento e Tratamento de Pedidos

Conforme Hirata (2002) de uma forma básica pode-se dizer que o servidor tem a função de receber requisições HTTP retornando respostas que podem ser desde páginas estáticas até resultados de buscas com conteúdo dinâmico.

Como já mencionado anteriormente, os servidores podem oferecer recursos que vão desde arquivos estáticos até *scripts* que podem gerar algum tipo de resposta mais elaborada. Após o recebimento do pedido, o mesmo é tratado em diferentes etapas:

1º Leitura e análise da mensagem: É feita a leitura da linha de requisição e verificada a operação solicitada pelo cliente (GET, POST ou outros métodos), bem como o URL do recurso em questão. O cabeçalho da requisição HTTP também é analisado para compor a mensagem de resposta.

2º Encontrar o arquivo: É feita a tradução do URL para encontrar o arquivo do recurso solicitado.

3º Verificar autorização: Caso o conteúdo solicitado seja de uso restrito, verificar informação de acesso que deve estar no cabeçalho da mensagem de requisição.

4º Geração e transmissão de resposta: O servidor verifica, por meio do sistema, o tamanho do arquivo, a última modificação, entre outros atributos importantes. Essas informações, juntamente com a identificação do servidor e a hora atual formam o cabeçalho de identificação do recurso. O servidor agrega então a Linha de *Status*, o cabeçalho e o próprio recurso e retransmite ao cliente.

Quando um cliente já possui o recurso em *cache* é feita apenas uma validação com o servidor através de comparação pela hora da última alteração realizada no arquivo.

A mensagem de resposta pode ser gerada de várias maneiras, podendo apenas apanhar arquivos associados ao URL e remeter ao cliente ou até mesmo a invocação de *scripts* e se comunicando com outros servidores e bancos, formular uma resposta (KRISHNAMURTHY;REXFORD, 2001).

4.4.2 Arquitetura

Um servidor Web está sujeito (e é o que normalmente acontece) a ter que trabalhar com várias requisições ao mesmo tempo. Os servidores são classificados conforme tratam esses pedidos, podendo ser Controlados por Evento, Controlados por Processo e até mesmo esquemas híbridos desenvolvidos na tentativa de aproveitar as qualidades destes dois tipos de arquitetura.

4.4.2.1 Servidores Controlados por Evento

Neste tipo de servidor existe um único processo que alterna entre o atendimento a diferentes pedidos. Não necessariamente o servidor aceita apenas um

pedido por vez, mas caso aconteça de ser aceito mais de um pedido, o processo alterna entre as tarefas de cada pedido, atendendo apenas uma de cada vez.

A grande maioria dos servidores Web não é controlado por evento já que são muitas as dificuldades encontradas no desenvolvimento deste *software*, como também as limitações do sistema operacional que não representam um suporte eficiente e confiável para este tipo de arquitetura (KRISHNAMURTHY;REXFORD, 2001).

4.4.2.2 Servidores Controlados por Processo

Neste tipo de arquitetura é dedicado um processo separado para cada pedido. Existe um processo mestre que fica escutando as novas conexões de clientes, este se bifurca em sub-processos, sendo assim permite que o servidor atenda vários pedidos ao mesmo tempo.

A grande maioria dos servidores Web optou por este tipo de arquitetura, onde a criação de multiprocessos trouxe um ganho significativo no desempenho dos mesmos, contudo, possui alguns problemas ou limitações que prejudicam o desempenho. O maior problema é o *Overhead*[6] que existe na passagem de um processo para outro. Muitas vezes, nesta passagem, o sistema operacional precisa salvar ou atualizar informações ou até mesmo buscar informações em disco, o que pode representar uma perda substancial no desempenho (KRISHNAMURTHY;REXFORD, 2001).

[6] Tempo durante o qual o sistema não está produzindo trabalho útil para qualquer usuário.

4.4.4 Servidor Apache

Em Junho de 2006 a NetCraft divulgou uma pesquisa apontando que cerca de 61,25% dos sites na Internet rodavam sobre o Apache, apontando o *software* como o Servidor mais utilizado no momento em todo o mundo. Isto se deve principalmente à gratuidade e boa documentação, com o código fonte completo e licença irrestrita. Sua implementação é feita em módulos, o que permite aos seus servidores se ajustarem em tempo de compilação da melhor maneira possível ao sistema operacional utilizado, sejam eles orientados a eventos ou processos. Suas principais qualidades são: multiplataforma, robustez, *performance* e adaptabilidade. É compatível com a especificação HTTP/1.1, permite mudanças em suas características (flexibilidade) mesmo em suas partes mais internas (*core*) através da utilização de módulos, e tem sua própria API padronizando toda programação interna (HIRATA, 2002).

Por *default*[7] normalmente a instalação do Apache vem com um formato bastante "genérico", sendo possível o ajuste de suas configurações para cada caso específico. Serão descritos a seguir os principais parâmetros a serem configurados na tentativa de extrair o máximo do servidor instalado.

4.4.4.1 *TypeOfServer*

Este parâmetro pode ser configurado como: "inetd" ou "standalone", sendo que quando setado com a opção "inetd" em servidores com uma alta carga de trabalho, degrada consideravelmente o seu desempenho, isto pelo fato de usar um aplicativo (inetd ou xinetd) como intermediário para suas conexões (HIRATA, 2002).

[7] Opção para a qual o programa já vem configurado para seu funcionamento caso o usuário não altere suas configurações.

4.4.4.2 *MinSpareServers, MaxSpareServers, e StartServers*

Responsáveis por controlar o número de processos inativos que estão rodando no Apache a fim de atender novos pedidos que venham a ser realizados. *MinSpareServers* determina um número mínimo e *MaxSpareServers* o máximo de processos inativos que devem ser mantidos. Um número muito baixo pode fazer com que quando muitas requisições forem iniciadas, estes processos se esgotem e o Apache tenha que criar processos filhos, e isto certamente terá um custo bastante elevado, já que, ao invés de estar disponível para atender às requisições, estará ocupado criando processos. Um excesso de processos inativos, ou seja, um valor muito alto para *MaxSpareServers* pode causar um consumo desnecessário de recursos da máquina. Estes parâmetros devem ser setados para um valor para que o Apache não precise freqüentemente abrir mais de 4 processos por segundo, sendo que este pode abrir, no máximo, 32 processos por segundo (RAM, 2003).

StartServers define quantos processos serão criados quando o Apache for inicializado. Possui pouca influência na performance do Servidor pelo fato de que o servidor é reiniciado esporadicamente e normalmente em horários especiais, porém, caso o número de requisições para o servidor seja muito elevado, deve-se setar um valor alto para que não perca muita performance na hora da partida. O Servidor criará processos até que o número de *MinSpareServers* seja atingido (RAM, 2003).

4.4.4.3 *ThreadsPerChild, MinSpareThreads e MaxSpareThreads*

Análoga à *StartServers* (plataforma multiprocesso) a diretiva *ThreadsPerChild* configura o número máximo de conexões simultâneas do Apache em plataformas orientadas a *Threads*, como é o caso do Windows. As tarefas são executadas por apenas um processo filho, isso quer dizer que este parâmetro definirá também o número máximo de requisições para o servidor. Existe também a diretiva *MaxThreads* que seria a verdadeira responsável pelo controle do número de *threads* a serem utilizadas, porém, só tem utilidade em plataformas multiprocessos onde a diretiva *ThreadsPerChild* não corresponde ao valor máximo de *Threads* a ser usado, caso contrário, basta setar com um valor acima do atribuído a *ThreadsPerChild* e deixar que esta variável faça todo o controle(HIRATA, 2002).

Os parâmetros *MinSpareThreads* e *MaxSpareThreads* definem os valores mínimo e máximo de tarefas inativas, assim como *MinSpareServers* e *MaxSpareServers* controlam os processos inativos do Apache.

4.4.4.4 *AllowOverride*

Esta diretiva serve para liberar o que o cliente poderá alterar em um determinado diretório e, conforme for setada, procurará em cada diretório que se tentar acessar pelo arquivo indicado pela diretiva *AccessFileName*, que por padrão é o arquivo .htaccess (RAM, 2003).

O arquivo .htaccess é um arquivo texto do Servidor Web Apache responsável pela restrição de acesso e autenticação de usuários ao servidor. Serve para proteger arquivos e diretórios que contenham informações restritas ou sigilosas,

contudo, deve se tomar cuidado quando utilizá-lo, pois pode afetar o desempenho do servidor (SANTOS, 2005).

4.4.4.5 *FollowSymLinks* e *SymLinksIfOwnerMatch*

A diretiva *FollowSymLinks* serve para que o servidor possa fornecer as informações aos clientes sem precisar checar se o arquivo em si, ou qualquer diretório acima deste é um link simbólico, processo que pode causar o aumento da latência do serviço.

Se a opção *SymLinksIfOwnerMatch* estiver setada, então o servidor permitirá acessar o link apenas se o destino dele bater com o dono do link, tendo assim que realizar consultas adicionais ao sistema para fazer esta validação.

Sendo asim, para se conseguir máxima performance, deve–se setar sempre *FollowSymLinks* em todos os diretórios, evitando o uso da opção *SymLinksIfOwnerMatch*, colocando como opção em casos realmente necessários e somento nos arquivos ou diretórios necessários (RAM, 2003).

4.4.4.6 *MaxRequestsPerChild*

Serve para configurar quantas conexões um processo filho pode atender antes de terminar, quando atinge este número então o processo é encerrado. Em sistemas onde não ocorre “vazamento de memória”, a melhor opção de configuração seria 0 (zero), ou seja, um processo poderia atender a ilimitadas conexões, contudo, caso o Apache venha a ocasionar um consumo crescente de memória este parâmetro deve

limitar as conexões a serem atendidas. Existem ferramentas que podem diagnosticar vazamentos de memória. Nestes casos deve-se também cuidar para não configurar um valor muito baixo o que poderia provocar a queda de desempenho do servidor, que ficaria criando e destruindo processos constantemente (RAM, 2003).

4.4.4.7 *MaxClients*

Sabe-se que a freqüência com que um servidor utiliza *swap*, ou seja, busca informações no disco rígido é muito importante no desempenho do mesmo, já que a leitura e escrita em disco são de 10 a 100 vezes mais lentas do que na memória RAM. Conforme Miranda (2007), o valor para *MaxClients* deve ser calculado da seguinte forma:

$$\text{MaxClients} = \frac{\text{RAM Total} - \text{QTMemOc}}{\text{XMemOcPF}}$$

Onde: **RAM Total**: Quantidade Total de Memória RAM.

QTMemOc: Quantidade de Memória Ocupada com o Servidor Desligado.

XMemOcPF: Média de Memória Ocupada por Processo Filho.

Obs.: O valor para *MaxClients* obtido nessa divisão indica o número máximo de clientes que o seu servidor pode atender sem acessar o disco rígido.

4.4.4.8 *KeepAlive* e *KeepAliveTimeout*

KeepAilve é a diretiva responsável por determinar se uma mesma conexão TCP pode ou não enviar múltiplas requisições, o que traz grandes benefícios,

principalmente quando se trata de páginas com múltiplas imagens ou objetos. Mantendo-se o *KeepAlive* em "On", apenas uma conexão pode ser capaz de transportar todos os objetos ou imagens, ao invés de criar uma conexão nova para cada uma delas (RAM, 2003).

Já a diretiva *KeepAliveTimeout* serve para indicar o tempo que uma conexão deve aguardar até a próxima requisição, sem ser finalizada. A orientação é para que se estabeleça um valor entre 2 a 5 segundos. Um valor muito alto poderia prender processos que poderiam estar atendendo a novos clientes.

4.4.4.9 *MaxKeepAliveRequests*

Tem a função de controlar o número de requisições a ser atendidas por uma conexão persistente antes de ser encerrada. O estabelecimento de novas conexões aumenta o *overhead* e por isso deve-se setar esta diretiva com um valor alto, contudo, não se deve especificar como 0 (zero) o que poderia abrir margem na segurança para ataques de negação de serviço (MIRANDA, 2007).

4.4.4.10 *SendBufferSize*

Diretiva responsável por determinar o tamanho do *buffer* de saída em conexões TCP. A cada conexão de um novo cliente, um *buffer* TCP é criado para enfileirar as informações pertinentes às suas requisições. È muito importante para o desempenho em servidores com uma grande carga de trabalho e com um grande número de conexões, porém, deve se tomar cuidado, pois um valor muito alto pode consumir muita memória, podendo causar prejuízos ao desempenho do servidor (HIRATA, 2002).

4.4.4.11 *Timeout*

Baseando-se no tempo entre estabelecimento e o recebimento da requisição em uma conexão e desde o último *ACK*[8] recebido, determina quanto tempo uma conexão será considerada ainda ativa pelo Apache e após isso finalizada. Isso não afeta conexões persistentes onde o tempo de vida de uma conexão é determinada por *KeepAliveTimeout* (HIRATA, 2002).

Este valor depende da conexão existente e do tempo de resposta que podem ser diferentes para cada caso, mas em geral o valor de 300 segundos, que é o valor default, é considerado muito alto, devendo-se reduzi-lo para 150 ou 75 segundos.

4.4.4.12 *ListenBacklog*

Diretiva responsável por determinar o tamanho da fila de recebimento de novas conexões TCP, tem como valor padrão 512. Para que se consiga alterar esse valor é necessário que a mudança seja realizada também no sistema operacional (HIRATA, 2002).

4.4.4.13 *LimitRequestBody*

Tem como função limitar o tamanho do corpo de uma requisição para os métodos PUT e POST para evitar problemas com ataques de negação de serviço. O tamanho máximo para uma requisição seria de 2GB e por *default* esta diretiva é setada como 0 (zero), o que indica que não foi estabelecido limite para esta variável, o que

[8] Indicador enviado pelo destinatário para o remetente de uma requisição afirmando que o segmento foi corretamente enviado.

pode gerar aberturas na segurança. Aconselha-se um valor em torno de 10 KB para garantir tamanho mais do que suficiente para as respostas dadas pelos usuários (HIRATA, 2002).

4.4.4.14 *LimitRequestFields*

Responsável por limitar a quantidade de informação presente nos cabeçalhos das requisições http. Apesar de o valor *default* vir como 100, a configuração com o valor 50, mesmo nas transações mais complexas, já possuirá uma margem de folga e não prejudicará em nada no atendimento das mesmas (HIRATA, 2002).

4.4.4.15 *Options Multiviews*

Útil apenas em casos onde sejam servidos documentos em múltiplas línguas, caso contrário não deve ser mantido, pois pode influenciar no desempenho do servidor. A configuração deve ser feita na seção *Directory* do arquivo de configuração (SILVA, 2006).

4.4.4.16 *FancyIndexing*

Apesar de permitir que o servidor mostre a listagem de conteúdo do diretório acessado no caso de o arquivo principal não ser encontrado, o comando deve ser evitado para não gerar mais carga do que o necessário. A não localização de conteúdo deve ser uma preocupação na hora de produzir os sites, e não como parte integrante da configuração do servidor (HIRATA, 2002).

4.4.4.17 *HostNameLookups*

Permite que o Apache registre a informação baseada no nome do cliente ao invés do endereço IP. Mesmo que mantenha essas resoluções de nomes na memória *cache*, consiste em um processo bastante custoso porque para cada novo cliente deve se fazer uma consulta ao servidor DNS, o que pode gerar tráfego desnecessário. Desta forma, é recomendável se desativar esta opção (RAM, 2003).

Quando se utiliza as diretivas *Allow from* ou *Deny from*, deve-se usar endereços IP, ao invés de nomes de domínios. Caso contrário uma consulta dupla ao DNS será executada para ter certeza de que o domínio ou host não está sendo *spoofado*[9].

4.4.4.18 *LogLevel*

Sabe-se que os *logs* do servidor representam uma fonte importante de informações para que se possa avaliar e fazer um planejamento adequado de acordo com as estatísticas de acesso, porém, a gravação dos *logs* é um grande responsável pelo consumo de recursos de CPU e tempo de disco, por esse motivo é muito importante que se avalie bem e que, se possível, descarte-se as informações que terão muito pouco ou nenhuma utilidade. Esta diretiva dos arquivos de configuração do servidor permite que se controle o *logging*[10] da forma mais conveniente (SILVA, 2006).

[9] Falsificação do endereço IP de origem nos pacotes transmitidos para esconder o endereço real ou personificar outro nó na rede.
[10] É o registro feito de um determinado log de acesso ou de erro do sistema.

4.4.4.19 *MMapFile*

Quando ativado, (recompilando o executável do Apache) permite especificar arquivos para serem mapeados em memória, mantendo-os disponíveis permanentemente, sem a necessidade de acessá-lo a partir do disco. Caso o arquivo seja atualizado será necessário reinicializar o Apache para buscar as novas informações no HD (HIRATA, 2002).

Esta opção só é aplicável a arquivos estáticos.

4.4.4.20 *Mod_bandwidth*

Este módulo do Apache permite controlar a quantidade de dados enviados por segundo, e baseando-se no IP e no tamanho do arquivo consegue assim dividir a largura de banda existente entre os seus clientes mantendo o serviço para todos, mesmo que aumente consideravelmente a latência (HIRATA, 2002).

5 AVALIAÇÃO DE DESEMPENHO

Além de eficaz, ou seja, realizar os serviços e funções designadas a ele e com toda a segurança, um sistema também deve ser eficiente e deve realizar os trabalhos propostos dentro de parâmetros de desempenho (tempo e processamento) aceitáveis. O trabalho deve ser realizado a fim de conseguir um melhor rendimento e satisfação por parte do usuário (MISAGHI, 2005).

5.1 PARÂMETROS DE DESEMPENHO

De uma forma bem genérica, pode-se dizer que o desempenho de um sistema é determinado por dois fatores principais: pela sua latência e pelo *throughtput*, mas é claro, nunca esquecendo do fator segurança.

5.1.1 Latência

Consiste no intervalo de tempo medido entre o início e o tempo de resposta de uma determinada requisição. O resultado é expresso em unidades de tempo, segundos ou milisegundos, por exemplo (PINHEIRO, 2004).

5.1.2 Throughtput

Faz uma análise inversa à da latência. O seu resultado consiste no número de itens processados por unidade de tempo, como por exemplo, bits transmitidos por segundo ou então operações http realizadas por dia (TAROUCO, 1999).

5.2 MEDIÇÃO E AVALIAÇÃO DO TRÁFEGO DA WEB

Até agora se demonstrou e descreveu brevemente sobre protocolos de rede básicos, principalmente o http, e também foi apresentada uma breve descrição do funcionamento dos servidores Web, isto porque estes são conceitos básicos para que se possa entender e começar a falar sobre medição da Web e avaliação de resultados destas medições.

5.2.1 Motivação da Medição

É de fundamental importância que se consiga realizar medições de tráfego da Web já que desta forma é possível verificar e caracterizar padrões de acesso dos usuários, o desempenho dos softwares da Web, bem como analisar a infra-estrutura da rede de suporte.

Do ponto de vista gerencial, permite criar estatísticas de acessos e carga de trabalho imposta a cada site e aos servidores Web em geral. Essas informações tornam-se importantes em tomadas de decisões, como na alocação de recursos, para fins de cobrança e auxiliam na detecção de problemas de desempenho.

Os servidores Web possuem uma grande quantidade de parâmetros a serem configurados e que afetam diretamente na locação de recursos. Uma medição adequada pode auxiliar na configuração destes parâmetros e com isso conseguir um melhor aproveitamento dos recursos e, conseqüentemente, ganho de desempenho (KRISHNAMURTHY; REXFORD, 2001).

5.2.2 Etapas de Medição e Avaliação de Tráfego

Existem três etapas principais para a medição e avaliação de tráfego. São elas: a monitoração das transferências da Web em algum local, a geração dos registros de medição em um formato específico e o pré-processamento dos registros como preparação para análise subseqüente (KRISHNAMURTHY; REXFORD, 2001).

5.2.2.1 Técnicas de monitoramento do tráfego da Web

Segundo Krishnamurthy e Rexford (2001), as técnicas de monitoramento podem ser classificadas da seguinte forma:

a) *Logging* do Servidor

Existem hoje, diversas ferramentas comerciais desenvolvidas que possuem a finalidade de analisar os *logs* gerados pelo servidor. Isto se deveu ao fato de que são poucos os fornecedores de servidor e os formatos de *log* gerados muito parecidos.

O servidor gera um *log* no tratamento de cada solicitação http. Esses *logs* possuem informações como a identificação do cliente solicitante, o horário da solicitação, o método do pedido, URI e o código (S*tatus*) da resposta. Essa fonte de informação representa a principal base de dados para a área de desempenho e tráfego da Web, porém, mesmo assim possui várias limitações. Não é possível registrar todo o cabeçalho do pedido pois criaria um aumento considerável no *Overhead*.

Os *logs* do servidor não registram validações de *cache*, então, a quantidade de acessos e solicitações de um cliente por um determinado recurso podem não

representar uma informação real. É claro que a operação de validação de *cache* pode ser evitada (quebra de *cache*), contudo, isso aumentaria consideravelmente o tráfego gerado. Além disso, existe uma grande dificuldade na associação de pedidos com o cliente, pois a atuação de *proxies*, atribuição dinâmica de IP, entre outros fatores, podem gerar em um mesmo cliente com vários endereços diferentes ou vários clientes com o mesmo IP. Uma solução para este problema poderia ser a utilização de *cookies* de identificação.

b) *Logging* do Proxy

Assim como nos servidores, os *proxies* geram *logs* no atendimento de pedidos. Sua grande vantagem é uma melhor identificação e informações sobre clientes devido à sua localização mais próxima, bem como também o registro do atendimento de pedidos pela própria *cache* do *proxy*. Contudo, este também não registra os pedidos atendidos pela *cache* do *browser* ou *proxies* mais próximos do cliente.

c) *Logging* do Cliente

O registro dos *logs* no próprio cliente permite caracterizar e representar um padrão mais aproximado para o comportamento real de navegações e utilização por parte do usuário.

Neste tipo de operação é possível verificar também todos os pedidos atendidos pela *cache* do *browser*, além de verificar transações interrompidas pelo usuário.

O maior problema neste caso é que os *browsers* não geram *logs* por *default* e para que fosse possível essa análise seria necessário um *browser* popular, com código aberto para fazer as modificações necessárias, além da distribuição desta versão a um número representativo de usuários. Isso, devido ao aumento de carga gerada, normalmente torna esta técnica inviável.

d) Monitoração de Pacotes

A grande vantagem desta técnica é o fato de não afetar nas funções e no desempenho do *browser* e dos servidores.

Consiste em uma forma de capturar pacotes IPs que trafegam em um determinado *link* ou por um roteador. Com um monitor é possível fazer um acompanhamento dos pacotes de solicitação e de resposta em função do tempo, permitindo verificar atrasos, perdas de pacotes e o *throughput* na conexão.

Permite a captura de cabeçalhos ou até pacotes internos, contudo, é necessário um suporte de hardware e software muito eficientes para a captura, processamento e armazenamento de informações, e quanto maior o tráfego, maior largura de banda, maior será a carga de trabalho deste monitor. Para que isso seja possível é preciso dividir o cabo ou um suporte no roteador para passar a cópia dos pacotes ao monitor.

Suas principais desvantagens se devem ao fato de estar restrito ao monitoramento de uma parte limitada da rede, não garantindo que todas as conexões entre os *hosts* serão por esta rota e, além disso, também não captura pedidos satisfeitos pela cache do browser ou informações criptografadas pelo *Secure Socket Layer* (SSL).

Questões de privacidade também impedem a utilização desta técnica, já que o monitor acessa cada pacote de informação desta conexão.

e) Medição Ativa

As técnicas anteriores possuem algumas limitações principais: a captura das mensagens é feita em único local, o monitoramento é feito sem controle de quando os pedidos ocorrem ou até mesmo se foram cancelados, sem contar que não analisam a latência de consulta ao DNS.

No caso da Medição Ativa para a análise do desempenho, a geração de pedidos é feita de forma controlada, não sendo apenas feita a coleta de informações aleatórias. Uma ferramenta empregada no *browser* (agente usuário) gera requisições com base em uma lista de URIs e horários em que estas devem ser feitas e registra o resultado das respostas obtidas.

São necessários que se analise três aspectos principais: onde colocar os clientes para medição, já que o resultado pode ser influenciado pelo conteúdo, pelo tráfego e pela largura de banda. A quem fazer os pedidos, já que se pode sofrer influência pela popularidade dos *sites*, ou até mesmo pela plataforma utilizada. E por fim, que dados se deseja obter com a medição.

5.3 FERRAMENTAS ESTUDADAS

Durante o decorrer de todo o trabalho foram estudas várias ferramentas que pudessem vir a auxiliar nas tarefas de avaliar os servidores e se possível o próprio cenário estudado como um todo. Foram selecionadas algumas destas conforme a

necessidade e também pelo fato de serem *freeware*[11]. A seguir será feita uma breve descrição de cada uma delas.

5.3.1 Webalizer

Representa uma poderosa ferramenta para análise de arquivos *log* de servidores, escrita em C consegue de forma muito rápida analisar milhares de registros, sendo compatível com diferentes tipos de formatos de *log,* entre estes, o formato comum utilizado pelo Apache (CISNEIROS, 2003). Seus resultados são apresentados na forma de gráficos em documentos HTML, gerando estatísticas de acesso ao servidor e as páginas nele contidas. Por este motivo representa uma ótima opção para análise do consumo de recursos em qualquer servidor Web (BERBERT, 2004).

Sua instalação e configuração são bastante simples e seu fonte está disponível gratuitamente, bem como o manual de instalação e configuração. Na verdade, basta baixar o programa e salvá-lo em alguma pasta do próprio servidor, configurar qual(ais) o(s) arquivo(s) de *log* a serem analisados e onde as estatísticas serão geradas, permitindo ou não o acesso via Web.

5.3.2 MRTG - Multi Router Traffic Grapher

Ferramenta *freeware* que utiliza o protocolo **SNMP** ou *scripts* externos para obtenção de informações de tráfego dos mais variados objetos presentes em um ambiente de rede. Apesar de ser uma ferramenta usada basicamente para análise de tráfego, quaisquer dados obtidos através do protocolo SNMP podem ser monitorados

[11] Programa de computador gratuito para o público.

por essa ferramenta. O MRTG gera gráficos que podem, como já foi mencionado, ser visualizados em formato HTML (Figura 9). Desenvolvido usando as linguagens de programação PERL e C, pode ser instalado no sistema operacional Linux ou Windows (OETIKER, 2007).

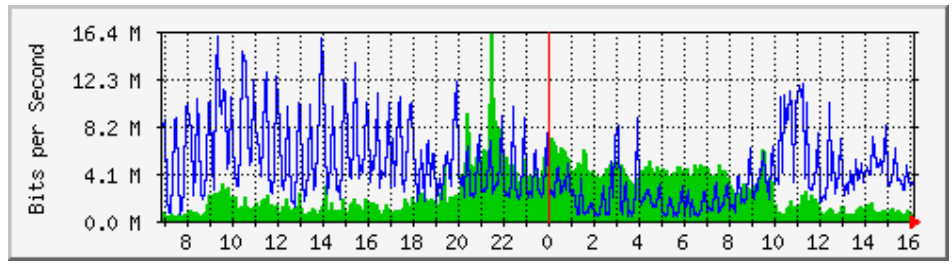

Figura 9.Exemplo de Gráfico de tráfego gerado pelo MRTG
Fonte: OETIKER, T. (2007)

O download do arquivo de instalação do MRTG pode ser feito da seguinte fonte: <http://people.ee.ethz.ch/~oetiker/webtools/mrtg/pub/>.

A instalação do programa é bastante simples, no entanto, a parte de configuração do mesmo exige mais conhecimento. É necessário configurar o equipamento informando onde estará rodando o MRTG, informar o diretório onde serão gerados os resultados estatísticos, criar o diretório MRTG para configurar a interface física, além ainda de configurar a CRONTAB para determinar a atualização dos dados e o arquivo de configuração principal. Contudo existem vários tutoriais de instalação e configuração do MRTG disponíveis na Internet e que podem auxiliar na utilização deste programa.

A partir daí os arquivos HTML já podem ser acessados via *browser* para visualizar as estatísticas

5.3.3 Nagios

Programa apresentado como um poderoso monitor de rede, sendo possível com este monitorar e atestar o funcionamento adequado dos equipamentos e serviços. Com ele é possível criar grupos de usuários para receber relatórios e alertas do sistema quando algum problema ou situação crítica venha a acontecer. O programa identifica graficamente cada ponto (*host*) monitorado, identificando em cores diferenciadas a situação atual. Por exemplo, o vermelho identifica os serviços que não estão funcionando (DOWN) ou em fase crítica (CRITICAL), o verde os que funcionam normalmente (OK) e o cinza os pendentes (PENDING), entre várias outras (GALSTAD, 2006).

O *download* do programa pode ser conseguido de forma fácil e gratuita do site <http://www.nagios.org>. Permite ser instalado tanto em plataformas Linux como também Windows. A configuração para um perfeito funcionamento de todas as suas ferramentas não é uma tarefa muito simples, entretanto existem inúmeros manuais de instalação e configuração disponíveis na Internet que facilitam nesta tarefa, mesmo para usuários menos experientes.

5.3.4 WAST - Web Application Stress Tool

Ferramenta que permite gerar carga de forma artificial para um servidor Web, possibilitando medir a capacidade de resposta, tanto em tempo (latência) como também na taxa de transferência (*throughput*) o que pode auxiliar na otimização de desempenho. Através de *scripts* formulados pode-se controlar e conhecer a carga a que está sendo imposta o servidor (MICROSOFT, 2007).

Para fazer a instalação do programa basta baixar o arquivo do programa (Setup.exe), que é *freeware*, e executá-lo. Pode-se encontrar no seguinte endereço: <http://www.microsoft.com/downloads/details.aspx?familyid=e2c0585a-062a-439e-a67d-75a89aa36495&displaylang=en>.

Para se criar um novo *script* basta escolher no menu, na parte superior da página a opção *Scripts*, *Create* e então escolher o tipo de *script* que será criado. O próximo passo seria então fazer as devidas configurações de *script*, indicando o servidor a ser testado e quais os tipos de requisição a serem feitas a fim de simular uma situação real de trabalho. Além disso são necessárias ainda outras configurações que se deseja para o teste, como o número de *threads* e o tempo de duração para o mesmo, dentre outras várias informações necessárias, dependendo de cada situação. Em seguida é só colocar o WAST a rodar o *script*.

Os resultados são gerados na forma de relatórios detalhados onde constam todas as informações relativas ao teste realizado como: quantidade de *hits*, taxa de envio e de recebimento(KB/s), quantidade de informações enviadas e recebidas(KB), erros ocorridos e, principalmente, a média para o tempo(ms) de resposta das requisições.

5.3.5 Dr. TCP

Programa que auxilia na hora de fazer alterações em registros do sistema relacionados às conexões TCP. Através de uma interface em forma de formulário consegue-se alterar valores para registros que controlam, por exemplo, o valor inicial para o tamanho da janela de recebimento (RWIN) em conexões desse tipo.

O *download* do programa pode ser feito no seguinte endereço: <http://www.dslreports.com/front/drtcp.html> e para fazer a instalação basta executar o arquivo sem ser necessário qualquer tipo de configuração.

A instalação e configuração do Dr. TCP é simples e rápida, porém, para se manipular registros do sistema é necessário que se tenha um conhecimento mais aprofundado, pois alterações indevidas podem prejudicar o funcionamento do sistema.

6 AVALIAÇÃO DE DESEMPENHO E OTIMIZAÇÃO DE SERVIDORES WEB APACHE

É notório que os motivos da queda de desempenho na Web e nos seus servidores está relacionada a uma série de fatores. Os problemas de atrasos nas respostas (latência) pode estar ligado não somente a problemas de capacidade de *hardware,* mas também a fatores como os protocolos utilizados, configurações do próprio servidor e sistema operacional, dentre outros fatores.

Assim, o objetivo idealizado para o projeto é estudar os fatores relacionados ao desempenho na Web, focando o software do servidor. Analisar cada um destes fatores e suas interações, apontando problemas ou limitações e possivelmente até mesmo sugerir soluções de melhorias para ganho de desempenho.

Para isto estão sendo estudadas e serão determinadas ferramentas de monitoramento de tráfego com as quais serão feitas as coletas de dados sobre a latência e *throughput* em um ambiente real e em simulações, dados que serão utilizados como parâmetros de análise de desempenho. A idéia é poder medir a condição atual, propor soluções ou otimizações para eventuais problemas encontrados e testar a efetividade das ações tomadas, além de comparar os resultados obtidos no ambiente real e simulado.

6.1 ESTUDO DE CASOS E ANÁLISE DE RESULTADOS

Em paralelo com o estudo de ferramentas que poderiam auxiliar nas etapas de diagnóstico e otimização de servidores Web, foram realizados vários testes em diferentes cenários com o intuito de demonstrar os problemas encontrados, dificuldades e até mesmo possibilidades de melhoria para cada situação.

6.1.1 Cenário1: Ambiente real

Uma empresa de grande porte, com uma grande e diversificada rede de computadores espalhada pelo país em várias filiais estava tendo problemas com reclamações vindas de seus próprios funcionários. Segundo estes, o sistema em alguns momentos estava se tornando muito lento, tanto Internet como também a Intranet.

Os servidores neste caso são dispostos separadamente, a Intranet fica em servidores internos da própria empresa, enquanto que a Internet fica em servidor terceirizado (alugado).

Quando uma reclamação era registrada, o problema era encaminhado ao próprio departamento de informática, este verificava a origem do mesmo, caso fosse interno o próprio operador do sistema ficava responsável por resolvê-lo, caso contrário, é aberta uma solicitação para a empresa contratada para os serviços de Internet. Tanto o registro das reclamações internas como as solicitações a terceiros são feitas por meio de ligações telefônicas.

Apesar de o problema ser facilmente resolvido, havia insatisfação por parte dos usuários, pois perdiam muito tempo enquanto ocorria a resolução do problema.

Iniciou-se então um estudo de ferramentas que pudessem auxiliar no gerenciamento desta rede de computadores, e que pudesse facilitar no diagnóstico de problemas e sua localização.

Primeiramente se pensou em fazer um estudo dos *logs* de acesso dos servidores para determinar quais eram os maiores vilões e que mais consumiam recursos da rede. Porém, chegou-se ao seguinte questionamento: o problema seria mesmo a banda que estaria muito sobrecarregada? Então decidiu-se fazer o seguinte:

1º Formatada e preparada uma máquina para que fosse utilizada para testar estas ferramentas;

2º Instalado e testado diferentes distribuições do linux, optando-se pelo Debian como o SO a ser utilizado, principalmente pela facilidade proporcionada para a instalação de novos softwares, onde o comando *apt-get install* baixa e instala os programas e todos os outros pacotes necessários;

3º As ferramentas disponíveis para gerenciamento testadas apresentam seus resultados e estatísticas na forma de páginas HTML por isso foi instalado o Apache como servidor Web, configurando-o para cada ferramenta a ser testada;

4º Primeiramente foi instalado o MRTG**,** funcionando perfeitamente, e surpreendentemente já foi possível verificar que o problema não está relacionado com a largura de banda, já que nos picos de utilização se chega a usar apenas 50% da banda disponível que é de 2 MB, o problema maior da rede não tem relação com a carga imposta, mas com os problemas que esta possui.

Com a finalidade de agilizar na detecção e resolução de problemas da rede, foi instalado então o NAGIOS, que através de uma interface gráfica permite ao operador visualizar o problema já no momento em que este ocorre, além de enviar mensagens eletrônicas de aviso diretamente para os responsáveis pela sua resolução. O programa encontra-se em fase de testes, mas representa uma ótima possibilidade de melhoria para o sistema e prestação de serviços.

Foram realizados também testes de monitoramento para o tempo de resposta para solicitações externas, onde se encontram problemas de largura de banda. O resultado obtido não foi nada satisfatório, chegando ao tempo de 68.052ms (68 segundos) para o carregamento completo da página quando submetido a uma carga de

40 *threads*, contra um resultado de 14.649ms obtidos no mesmo teste aplicado à página do *Google*. Testes realizados em uma conexão ADSL BRTurbo 250.

Ficava óbvio então que para clientes e usuários externos o resultado não estava agradando e seria necessário tomar alguma providência. O problema apontava para a quantidade de informação do site em relação à banda de Internet usada pelos usuários externos. Então, ou se diminuía o tamanho do site ou se arranjava uma forma para aumentar a taxa de transferência de dados. Havendo resistência contra a idéia de deixar o site mais simples e com menos efeitos de imagem e animação, o jeito seria compactar as informações requisitadas antes do envio.

O melhor método encontrado para amenizar o problema foi aplicar as seguintes linhas de comando:

```
<?
ob_start("ob_gzhandler");

// Conteúdo da página, pode conter código
// HTML, PHP,etc...

ob_end_flush();
?>
```

Este comando ativa e desativa o *buffer* de saída indicando o método de compressão a ser utilizado (“ob_gzhandler”).

Os resultados obtidos foram bastante significativos, baixando o tempo para a carga completa da página para 15.651ms (15 segundos).

Tabela 1. Evolução na latência

Condição	Latência (ms)
Normal	68.052,76
Comprimido	15.651,70

Neste caso a aplicação foi feita diretamente no código fonte do *site* pelo fato de não se ter acesso aos arquivos de configuração do servidor, contudo, pode se habilitar a compressão de dados pelo próprio servidor.

6.1.2 Cenário2: Ambiente Simulado

6.1.2.1 Descrição

Foi instalada uma rede com 3 computadores, sendo um deles configurado como servidor (PC1) e os outros dois como clientes (PC2 e PC3). Utilizou-se um Hub (8 portas), um modem conectado à Internet (ADSL) e cabos par trançado para fazer as conexões.

Tabela 2. Hardware utilizado para os testes de desempenho

	PC1	PC2	PC3
Sistema Operacional	Windows XP	Windows XP	Kurumin 5
Funcionalidade	Servidor	Cliente	Cliente
Processador	DualCore Intel Pentium 2666 MHz	AMD Athlon 900 MHz	AMD K6 II 466 MHZ
Placa Mãe	Asus P5V800-MX (Audio, Video, LAN)	PCChips M810LR (Áudio, Vídeo, LAN)	PCChips M810LR
Disco rígido	Maxtor 80 GB (7200 RPM)	ST315310A 15 GB (7200 RPM)	Maxtor 20GB
Memória RAM (MB)	1024	128	128

As informações referentes à rede de computadores instalada foram extraídas com a utilização do software Everest Ultimate Edition 2006 4.00.975.

O software escolhido para ser utilizado como servidor Web, foi o Apache por ser o mais conhecido atualmente e também por ser distribuído gratuitamente. Além disso, sua arquitetura modular permite um perfeito ajuste tanto a plataformas Linux como também Windows.

No PC2, configurado como cliente, foi instalado o software Web Application Stress Tool (WAST), ferramenta esta que seria utilizada para gerar diferentes cargas de trabalho para o servidor.

O PC3, também configurado como cliente, ficou como estação de trabalho normal, possibilitando o acesso à Internet e outras atividades durante os testes de monitoramento do servidor.

6.1.2.2 Etapas

O Apache por ser *open source*[12] possui inúmeras fontes e pacotes de instalação. A escolha da versão e dos pacotes do servidor Apache a serem baixados e instalados podem ter uma grande influência no desempenho do servidor já que por *default* podem vir configurados de uma forma bem genérica ou bem específica para algum determinado caso.

Para se conseguir o máximo de performance do servidor é preciso ajustá-lo para cada caso, o que pode não ser uma tarefa simples, porém, o ganho de desempenho pode ser muito grande.

Resolveu-se então partir de uma condição considerada pouco adequada, e estudar, testar e ajustar as diretivas dos arquivos do servidor e sistema operacional para avaliar a influência de cada uma delas sobre o desempenho do servidor. Para a configuração inicial do Apache foram tomados como base os arquivos httpd.conf disponíveis em <http://www.netexplorer.dk/APACHE/apache-02.php> e <http://stuff.mit.edu/afs/sipb/machine/anxiety-closet/configs/apachessl/httpd.conf>, contudo apenas de forma orientativa.

[12] Software também chamado de código aberto, ou seja, código fonte visível a todos os usuários.

A tabela 3 indica a configuração inicial para as principais diretivas relacionadas ao desempenho em um servidor.

Tabela 3. Diretivas de configuração

Diretiva	Valor
ThreadsPerChild	50
MinSpareThreads	5
MaxSpareThreads	50
MaxThreads	50
KeepAlive	off
KeepAliveTimeout	15
MaxKeepAliveRequests	50
Timeout	300
Options	Multiviews + SymLinksIfOwnerMatch
HostNameLookups	on
Loglevel	debug
MaxRequestPerChild	5
AllowOverride	all
AccessFileName .htaccess	habilitado

Como ponto de partida foi feito o teste de funcionalidade e o teste de carga no servidor para diferentes cargas de trabalho. Quanto à funcionalidade estava tudo certo e a página foi aberta pelo *browser* sem nenhum problema. A Tabela 4 mostra os resultados obtidos quando submetido ao teste de performance.

Tabela 4. Resultados de monitoramento obtido

Threads	20	40	80	160	200	250	280	300
Bytes Recv Rate (in KB/s)	13,28	43,85	43,95	43,89	44,08	7,62	6,95	5,94
TTLB Avg (ms)	684,07	431,13	862,22	1724,99	2148,79	15017,67	18326,37	22909,77

Os resultados obtidos neste ambiente de simulação não podem ser comparados a taxas e resultados obtidos no monitoramento de servidores dispostos na Internet, já que se trata de um ambiente isolado e com condições favoráveis de comunicação de dados entre as estações.

As otimizações foram feitas, então, em várias etapas diferentes:

a) Etapa 1

Nesta primeira etapa foi ativada (“On”) a diretiva *KeepAlive* que anteriormente estava setada como “Off” para que uma mesma conexão TCP pudesse fazer múltiplas requisições.

Como conseqüência da ativação da diretiva *KeepAlive*, é necessário configurar quanto tempo uma conexão ficaria aberta esperando uma próxima requisição através da diretiva *KeepAliveTimeout*. Um valor muito elevado pode provocar consumo desnecessário de recursos que poderiam estar atendendo a novas requisições. Este valor foi baixado de 15 segundos para 5 segundos. Após estas alterações, novas medições foram realizadas.

Tabela 5. Resultados otimização – Etapa 1

Threads	20	40	80	160	200	250	280	300
Bytes Recv Rate (in KB/s)	31,86	31,99	35,67	30,14	29,22	7,05	6,13	5,41
TTLB Avg (ms)	267,50	592,08	1060,75	2515,73	3236,81	16225,77	20748,23	25134,73

Neste caso notou-se uma sensível melhoria para uma carga imposta de 20 threads, porém para cargas maiores o resultado ficou ainda pior aumentando a latência e baixando a taxa de recebimento de dados. Como a partir deste ponto foi ativada a manutenção de conexões permanentes e o número de tarefas foi limitado nos arquivos de configuração, é provável que seria necessário o aumento destes parâmetros para que fosse possível o aumento do atendimento de tarefas simultaneamente.

b) Etapa 2

Foram feitas as configurações relativas ao número de tarefas a serem atendidas pelo servidor através das diretivas *ThreadsPerChild*, *MinSpareThreads*,

MaxSpareThreads, *MaxThreads*. Também alterado o valor de *MaxKeepAliveRequests* para aumentar o número de requisições a serem atendidas até que as conexões fossem encerradas, conforme demonstrado na tabela 6.

Tabela 6. Dados de configuração - Etapa 2

Diretiva	Valor Anterior	Valor Atribuído
ThreadsPerChild	50	250
MinSpareThreads	5	25
MaxSpareThreads	50	250
MaxThreads	50	1000
MaxKeepAliveRequests	50	100

Devido às configurações efetuadas o servidor passou a aceitar e manter um número maior de conexões abertas. Para cargas não muito elevadas, neste caso, até 160 threads, os resultados de desempenho ficaram piores, possivelmente porque isso possa ter causado um consumo desnecessário de recursos por parte do servidor que mantinha conexões abertas durante algum tempo sem que elas fossem mesmo necessárias, porém, em casos onde a carga de trabalho foi maior, houve um significativo avanço no desempenho, como se pode visualizar pela análise da tabela 7.

Tabela 7. Resultados otimização – Etapa 2

Threads	20	40	80	160	200	250	280	300
Bytes Recv Rate (in KB/s)	30,24	23,25	22,65	28,47	35,93	16,28	16,07	16,68
TTLB Avg (ms)	282,67	814,96	1673,14	2658,49	2636,77	7010	7955,96	8196,8

c) Etapa 3

Para o caso específico foi excluído a opção *Multiviews*, não sendo necessário servir documentos em múltiplas línguas, o que permite ganhar performance, evitando transações desnecessárias entre *browser* e servidor. Foi configurado *HostNameLookups* para “off” para que não fosse necessário fazer uma consulta ao

servidor DNS a cada conexão diferente. A diretiva *Loglevel,* que havia sido configurada para “debug”, resultando em arquivos de erro enormes, provocando muito consumo de recursos de processador e acesso a disco, foi passada para “crit”, para que se fizesse logging somente do que fosse extremamente necessário. No caso de utilização do Windows XP e que consiste em uma plataforma *multithreads*, como neste caso, recomenda-se setar a diretiva *MaxRequestPerChild* para 0, sendo que estava com o valor 5. Com estas novas configurações conseguiu-se uma enorme evolução de performance, como pode ser observado na tabela 8 pelas medições realizadas através do WAST.

Tabela 8. Resultados otimização – Etapa 3

Threads	20	40	80	160	200	250	280	300
Bytes Recv Rate (in KB/s)	139,38	457,79	423,61	439,88	408,52	418,2	425,2	425,41
TTLB Avg (ms)	33,99	38,67	86,31	168,52	228,27	278,93	306,87	328,72

Com estas novas configurações a latência realmente melhorou e percebeu-se que o gargalo para atendimento das requisições deixou, neste momento, de ser, em função da emissão, transmissão e recebimento de dados, mas passou a ser problema de processamento, observando-se através do gerenciador de tarefas que o processador trabalhava quase que constantemente em 100% de sua capacidade. Seria necessário então buscar desalocar recursos que poderiam estar sendo usados desnecessariamente.

d) Etapa 4

Foram substituídas então as diretivas “*SymLinksIfOwner*” por “*FollowSymLinks*”, e *AllowOverride* que estava setado como “all”, foi passado para

"none", então não seria mais necessário que o servidor verificasse *links* simbólicos para cada arquivo e diretório acessado e nem verificar restrições de acesso.

No ambiente virtual criado estas configurações estão se baseando apenas em sua influência sobre o seu desempenho, entretanto, em casos reais deve-se tomar cuidado ao trabalhar com a diretiva *AllowOverride* por questões de segurança para que não se fique exposto a ataques externos.

Tabela 9. Resultados otimização – Etapa 4

Threads	20	40	80	160	200	250	280	300
Bytes Recv Rate (in KB/s)	137,7	566,09	544,18	509,99	490,07	475,49	452,1	443,62
TTLB Avg (ms)	33,98	24,57	60,06	137,62	181,49	236,31	279,68	306,2

Neste caso pode se observar melhorias não só nos resultados de latência obtidos pelo WAST, mas também pelo uso do processador que caiu para valores próximos de 70% e pelo aumento da utilização de rede indicado pelo gerenciador de tarefas do Windows que também se elevou de 9% para 11% aproximadamente.

e) Etapa 5

A próxima etapa foi desabilitar a diretiva *AccessFileName* que indica o arquivo responsável pelas restrições de acesso para os usuários. Contudo, não houve uma melhora considerável conforme pode ser visto pelos valores obtidos pela simulação feita.

Tabela 10. Resultados otimização – Etapa 5

Threads	20	40	80	160	200	250	280	300
Bytes Recv Rate (in KB/s)	137,68	566,66	551,44	513,94	498,78	476,02	454,65	445,46
TTLB Avg (ms)	33,98	24,57	59,24	136,54	178,31	236,08	278,13	304,96

Isso deve ser explicável pelo fato de que a diretiva *AllowOverride* já está configurada como “none”, portanto realmente não representa uma mudança válida no que diz respeito a melhorias de performance.

f) Etapa 6

Durante esta etapa foram realizadas algumas otimizações no que diz respeito ao sistema operacional, buscando desabilitar todas as tarefas e serviços executados desnecessariamente ao funcionamento do servidor. Com a finalização destes serviços os processos executados simultaneamente pelo sistema operacional caíram de 36 para 30 processos. Também foi desativada a reserva de 20% da banda disponível para uso exclusivo feita pelo sistema operacional, além de diminuir o tamanho da lixeira que era de 10% do HD para 5%. Claro que sabendo que isto não tem um efeito imediato na performance do sistema, porém se levado em conta que apenas durante o período de testes realizados para este trabalho os arquivos de *log* de acesso e erro consumiram aproximadamente 35 GB de espaço de um total de 80 GB disponível do HD pode se prever que em uma condição real de trabalho o aproveitamento máximo de disco será muito importante.

Tabela 11. Resultados otimização – Etapa 6

Threads	20	40	80	160	200	250	280	300
Bytes Recv Rate (in KB/s)	137,33	566,95	551,67	510,64	488,70	476,43	453,73	445,03
TTLB Avg (ms)	33,99	24,53	59,21	137,44	181,98	235,87	278,67	305,20

Não houve alterações significativas nos resultados.

g) Etapa 7

Para se calcular um valor aproximado para a diretiva *ThreadsPerChild* foi utilizada a mesma fórmula que seria utilizada para calcular a diretiva *Maxclients* para plataformas multiprocessos como é o caso do Linux. Porém, o tamanho de uma página estática ou dinâmica pode ter uma variação muito grande. O Linux possui o comando "top" com o qual se pode conseguir o tamanho do processo, mas neste caso foram feitos uma série de testes para encontrar a melhor condição experimentalmente, ou seja, alterando as diretivas e testando. Os melhores resultados foram obtidos com valores aproximados de 350, cujo valor ficou atribuído à diretiva. Para que se consiga alterar este número acima de 256 é necessário alterar o parâmetro HARD_SERVER_LIMIT também para um valor superior, o que em servidores com uma demanda muito alta é bastante comum.

Algumas alterações também foram realizadas nos registros do sistema para tentar melhorar a performance TCP/IP do Windows XP. Utilizando o software Dr. TCP foram ajustados alguns parâmetros que podem afetar o desempenho que são: a Unidade Máxima de Transmissão (MTU), o valor inicial para o tamanho da janela para conexões TCP indicado pela variável RWIN e o tempo em segundos durante o qual o pacote é considerado ativo TTL, que foi setado para 64 segundos, apesar de não influenciar diretamente no desempenho. Isso será controlado diretamente nos arquivos de configuração do Apache. Para determinar o valor MTU deve se aplicar o comando ***Ping –f –l 1400 www.endereçoexterno.com.br*** e ir incrementando este valor (1400) até chegar ao ponto em que se encontre o gargalo da rede e se tenha a seguinte resposta "Packet needs to de fragmented but DF set". O valor imediatamente inferior a este será

o valor máximo para se enviar pacotes sem precisar fragmentar. Então somando mais 28 bytes a este valor correspondentes ao cabeçalho TCP/IP será obtido o valor MTU.

Para identificar o valor correto para RWIN é necessário que já se tenha calculado o valor correto para o tamanho máximo de segmento ou *Maximum Segment Size* (MSS), que pode ser obtido pela fórmula:

MSS = MTU – 40

Onde, 40 representa o tamanho do cabeçalho do pacote de dados em bytes.

O valor de RWIN deve ser sempre múltiplo de MSS e caso exceda o valor padrão de 65535 deve-se ativar as opções *Tcp1323Opts* ou *Windows Scaling*.

RWIN = 44 x MSS

Os valores obtidos pelo WAST após estas otimizações estão apresentados na tabela 12.

Tabela 12. Resultados otimização – Etapa 7

Threads	20	40	80	160	200	250	280	300
Bytes Recv Rate (in KB/s)	137,68	572,69	544,57	512,19	494,99	472,40	477,83	463,99
TTLB Avg (ms)	33,98	24,28	59,99	137,01	179,67	237,85	264,58	292,74

Considerando-se todos os testes realizados com diferentes cargas de trabalho impostas ao servidor, notou-se apenas pequenas melhorias.

h) Etapa 8

Buscando conseguir ajustar ainda melhor o tempo de resposta no servidor foram ainda ajustadas a diretiva *Timeout* de 300 para 75 segundos, *MaxKeepAliveRequests* de 100 para 1000 requisições e *KeepAliveTimeout* de 5 para 3

segundos buscando otimizar a utilização de recursos do servidor, evitando conexões inativas.

Tabela 13. Resultados otimização – Etapa 8

Threads	20	40	80	160	200	250	280	300
Bytes Recv Rate (in KB/s)	137,66	582,19	553,75	515,84	499,27	475,98	465,37	459,37
TTLB Avg (ms)	33,98	23,88	58,99	136,02	178,12	236,11	271,71	295,65

Observando-se em todas as diferentes cargas impostas ao servidor que de uma forma geral a resposta a essas novas configurações foi positiva, deve-se então mantê-las como definitivas.

Ao final dos testes pôde-se constatar a importância de uma correta configuração do Apache para cada caso específico, e que o fato de um servidor estar funcionando, não significa que ele esteja sendo eficiente, já que os usuários podem não estar satisfeitos com o serviço e acabarem indo para os sites da concorrência. Os resultados finais podem ser melhor visualizados em gráficos de latência e *throughput* montados a partir dos dados gerados pelo WAST, conforme figuras 20 e 21.

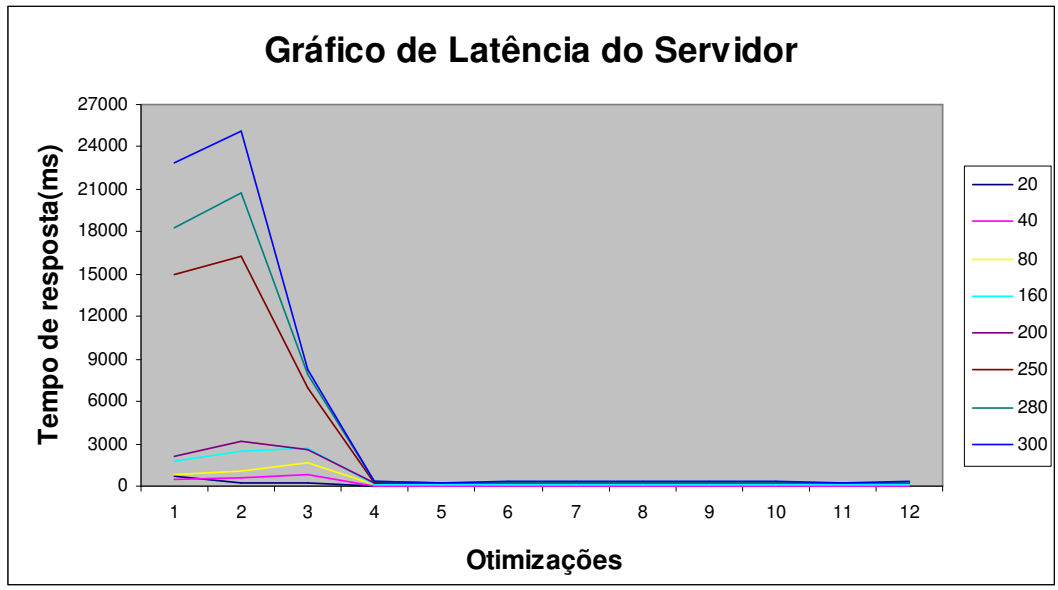

Figura 10.Gráfico da queda de latência devido a otimizações realizadas no servidor

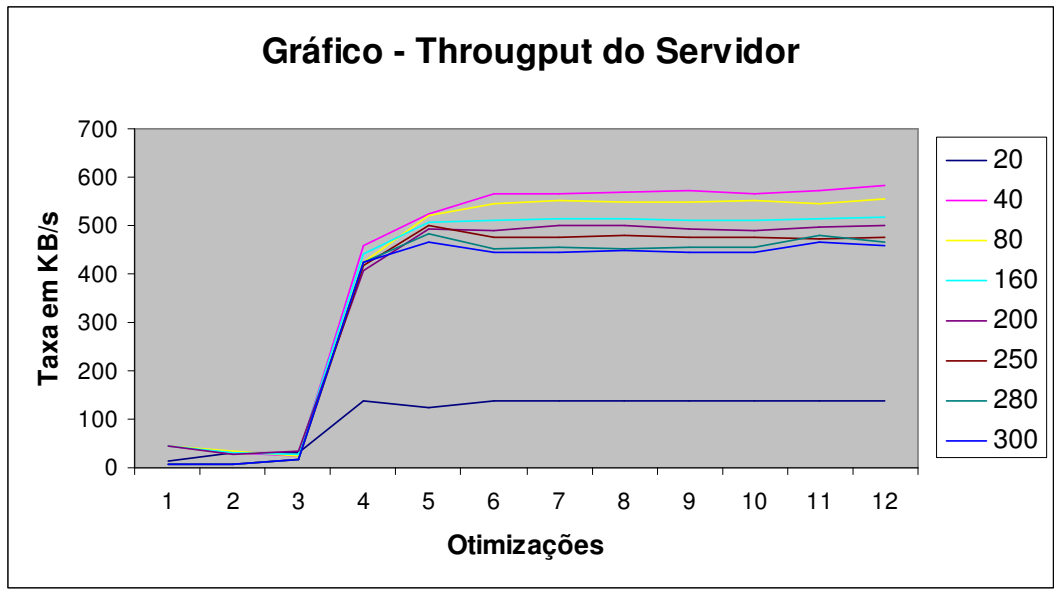

Figura 11.Gráfico da taxa de transmissão de dados frente a otimizações realizadas no servidor

É claro que se deve levar em consideração que neste caso se partiu da pior condição de configuração encontrada e que nas versões mais recentes algumas destas já vêm implantadas por *default*, porém, quando se trata de configuração de servidores Web, cada caso específico possuirá ajustes diferenciados, o que torna muito importante conhecer o funcionamento e a função de cada uma das diretivas existentes. Neste exemplo, para a pior situação chegou-se a latências por volta de 26 segundos e taxas de transmissão abaixo de 50 KB/s, apesar de que quando realizados os testes de funcionalidade tudo estava certo, e após todas as alterações e validações chegou-se a latências abaixo de 0,3 segundos e *throughput* se mantendo por volta de 450 e 550 KB/s para quase todas as cargas testadas.

CONCLUSÃO

A crescente e acelerada demanda por recursos disponíveis da Web aliado ao desconhecimento por parte dos usuários das operações envolvidas acaba resultando em um mau aproveitamento desses recursos e em uma Internet lenta e vulnerável.

Existe uma infinidade de softwares servidores e ferramentas para Web, contudo, as informações ainda são esparsas e complicadas o que torna difícil para usuários comuns conseguirem tirar o máximo proveito dos mesmos.

Os programas tornaram-se genéricos, de simples instalação e utilização, o que facilitou para que qualquer usuário seja capaz de colocar em funcionamento um servidor web, no entanto às custas de desperdícios de recursos que poderiam ser melhor aproveitados. Além disso, muitos profissionais esquecem de fazer uma avaliação externa, ou seja, do ponto de vista do cliente que pode não estar satisfeito com os serviços prestados. Por exemplo, o caso real que foi citado no Capítulo 6 deste trabalho demonstrou que uma avaliação feita internamente com resultados de compressão de dados não teriam relevância, porém quando avaliado em uma rede com largura de banda restrita teve uma grande importância, diminuindo muito a latência dos dados.

Existem ferramentas disponíveis gratuitamente que podem ser de grande valia na obtenção de dados para que se possa conhecer o tráfego e detectar problemas na rede. Conhecendo o comportamento dos usuários, bem como o tipo de informação que é enviada é possível parametrizar o software servidor de modo a se manter os níveis de segurança e confiabilidade necessárias sem prejudicar a viabilidade do sistema.

Neste caso, foi trabalhado basicamente sobre configurações de Software servidor e conexões TCP, e já se verificaram resultados expressivos, como pode ser vizualisado nos gráficos gerados de latência e *throughput*, contudo, ainda existem

diversas outras situações de otimizações por meio de *scripts* para buscas dinâmicas, otimização de acesso a informações de banco de dados, estruturação correta de informações e *sites*, além de outros diversos fatores que são importantíssimos e merecem um estudo e documentação detalhada que poderiam vir a ser fonte de informação para muitos usuários do setor.

REFERÊNCIAS

BATTISTI, Júlio. **Tutorial de TCP/IP**. Disponível em: <http://www.juliobattisti.com.br/artigos/windows/images/ncamadas-1.gif>. Acesso em: 10 ago. 2006.

BERBERT, Wanderson. **Criando relatórios estatísticos com o webalizer**. Disponível em:<http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=78>. Acesso em: 12 jan. 2007.

CANTU, Evandro. **Rede de Computadores e Internet**. Disponível em: <http://www.das.ufsc.br/~montez/Disciplinas/materialRedes/ApostilaCantu.pdf>. Acesso em: 17 jan. 2007.

CASAD, Joe; WILLSEY, Bob. **Aprenda em 24 horas TCP/IP**. Tradução de Daniel Vieira. Rio de Janeiro: Campus, 1999.

CISNEIROS, Hugo. **Montando Estatísticas com o Webalizer**. Disponível em: <http://www.devin.com.br/eitch/webalizer>. Acesso em: 13 nov. 2006.

COLOMBO, Flaviane Valvassori. **Ferramentas para Gestão de Conteúdo na Web**. Criciúma: UNESC, 2002.

FERREIRA, Bernardo Martins Vieira Coelho. **Sockets**. Disponível em: <http://artigos.com/artigos/tecnologia/sockets-1169/artigo/>. Acesso em: 05 jan. 2007.

FILHO, Ney Figueirôa de Senna; DIAS, Andrey Santana da Rocha; CRUZ, Luciano Brandão. **TCP Sobre ATM**. Disponível em: <http://www.rnp.br/newsgen/9909/tcp_atm.html>. Acesso em: 10 ago. 2006.

FILHO, Roberto Alves Gallo; BIANCHINI, Fernando Gobbi. **Aplicações para Internet**. Disponível em: <http://www.ic.unicamp.br/~geovane/mo410-071/Ch07-Resumo.pdf>. Acesso em: 10 set. 2006.

FOUCES, Oscar Dias. **A Localização de Páginas da Internet na Formação de Tradutores**. Disponível em: <http://www.confluencias.net/n1/fouces.html>. Acesso em: 20 abr. 2007.

GALLO, Michael A.; HANCOCK, Willian M. **Comunicação entre Computadores e Tecnologias de Redes.** Tradução de F.S. Correa da Silva. São Paulo: Thomson, 2002.

GALSTAD, Ethan. **Nagios**. Disponível em: <http://www.nagios.org/about/>. Acesso em: 10 ago. 2006.

HIRATA, Renato. **Otimizando Servidores Web de Alta Demanda**. Campinas: UNICAMP, 2002.

JUNIOR, Renato. **Apostila de Internet e Arquitetura TCP/IP.** Disponível em: <http://www.rjunior.com.br/downloads/tcp.pdf>. Acesso em: 5 mai. 2007.

KRISHNAMURTHY, Balachander; REXFORD, Jennifer. **Redes para a Web.** Tradução de Daniel Vieira. Rio de Janeiro: Campus, 2001.

KUROSE, James F.; ROSS, Keith W. **Redes de Computadores e a Internet: uma abordagem top-down**. Tradução de Arlete Simille Marques. 3. ed. São Paulo: Pearson Addison Wesley, 2006.

LIMA, João Paulo de. **Administração de Redes Linux – Passo a Passo**. 1. ed. Goiânia: Gráfica Terra Ltda, 2003.

LIMA, Michele Mara de A. Espíndula**. Métricas para a Internet**. Disponível em: <http://www.rnp.br/newsgen/9805/metricas.html>. Acesso em: 20 mar. 2006.

MARTINS, Jurandy. **Camada de Aplicação**. Disponível em: http://www.ic.unicamp.br/~jurandy/fac/aula-02.ppt. Acesso em: 3 fev. 2007.

MICROSOFT. **Web Application Stress Tool**. Disponível em: <http://www.microsoft.com/downloads/details.aspx?familyid=e2c0585a-062a-439e-a67d-75a89aa36495&displaylang=em>. Acesso em: 05 mar. 2007.

MIRANDA, Marco Aurélio Caldas. **Tunando o Apache 1.3.x ou 2.x (prefork)**. Disponível em: <http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=6104>. Acesso em: 08 mai. 2007.

MISAGHI, Mehran. **Análise de Desempenho de Redes**. Disponível em: <http://www.vision.ime.usp.br/~mehran/ensino/adr.html>. Acesso em: 20 jun. 2006.

OETIKER, Tobias. **MRTG Index Page**. Disponível em: <http://www.stat.ee.ethz.ch/mrtg>. Acesso em: 17 abr. 2007.

OETIKER, Tobias. **Tobi Oetiker's MRTG - The Multi Router Traffic Grapher**. Disponível em: <http://oss.oetiker.ch/mrtg/>. Acesso em: 14 jan. 2007.

PINHEIRO, José Maurício Santos. **Dispersão, Jitter e Latência**. Disponível em:<http://www.projetoderedes.com.br/artigos/artigo_dispersao_jitter_latencia.php>. Acesso em: 11 out. 2006.

RAM, Vishnu. **Configurando Apache para Performance Maxima.** Disponível em: <http://under-linux.org/wiki/index.php/Tutoriais/Apache/Apache-performance>. Acesso em: 27/05/2007.

RESELLER. **Como otimizar o desempenho do Unix na era da Internet**. Disponível em: <http://www.resellerweb.com.br/resellerschool/artigo.asp?id=12690>. Acesso em: 08 mai. 2007.

SANTOS, Rodrigo. **Teoria das Redes TCP/IP**. Disponível em: <http://forumweb.com.br/artigos/artigos.php?action=file&id=286>. Acesso em: 10 jun. 2006.

SANTOS, Victor Batista da Silva. **Restrição de Acesso á Servidores Apache Baseada em Autenticação por Senha**. Disponível em: <http://www.gris.dcc.ufrj.br/tutoriais/GRIS-2005-T-002.pdf>. Acesso em: 25 mai. 2007.

SILVA, Gleydson Mazioli da. **Apache**. Disponível em: <http://focalinux.cipsga.org.br/guia/avancado/ch-s-apache.htm>. Acesso em: 15 jan. 2006.

SOUZA, Lindeberg Barros de. **Redes de Computadores:** Dados, Voz e Imagem. Rio de Janeiro: Érica, 2001.

TAROUCO, Liane. **Projeto da Rede Metropolitana da Grande Porto Alegre**. Disponível em: <http://penta2.ufrgs.br/metropoap/ppt/curitiba/tsld033.htm>. Acesso em: 10 jan. 2007.

ZOTTO, Ozir Francisco Andrade. **Protocolo HTTP**. Disponível em: <http://www.pr.gov.br/batebyte/edicoes/1996/bb57/protocol.htm>. Acesso em: 14 nov. 2006.

BIBLIOGRAFIA RECOMENDADA

ALBUQUERQUE, Fernando. **TCP/IP Internet: Programação de Sistemas Distribuídos Usando HTML, JavaScript e Java**. Brasília: Axcel Books, 2001.

ALVES, Maria Bernardete; ARRUDA, Suzana Margareth. **Como Fazer Referências: bibliográficas, eletrônicas e demais formas de documentos**. Disponível em: <http://bu.ufsc.br/framerefer.html>. Acesso em: 06 mar. 2006.

COMER, Douglas E.; STEVENS, David L. **Interligação em rede com TCP/IP**. Tradução de Ana Maria Neto Guz. Rio de Janeiro: Campus, 1999.

DANESH, Arman. **Dominando o Linux.** Tradução João Eduardo N. Tortello. São Paulo: Makron Books, 2000.

MAXWELL, Scott. **Kernel do Linux.** Tradução de Carlos Mink. São Paulo: Makron Books, 2000.

REICHARD, Kevin. **Servidor Internet com Linux.** Tradução de Marcos Vieira. São Paulo: Berkeley,1998.