UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

ROBSON VENSON

SIMULAÇÃO DA CARACTERIZAÇÃO DA CARGA DE TRABALHO EM UM SERVIDOR WEB

CRICIÚMA, NOVEMBRO 2009

ROBSON VENSON

SIMULAÇÃO DA CARACTERIZAÇÃO DA CARGA DE TRABALHO EM UM SERVIDOR WEB

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

Orientador: Prof. MSc. Paulo João Martins

CRICIÚMA, NOVEMBRO 2009

ROBSON VENSON

Simulação da Caracterização da Carga de Trabalho em um Servidor Web

Submetido ao corpo docente do Curso 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. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

Prof. MSc. Paulo João Martins (UNESC)
Orientador

Prof. MSc. Rogério Antônio Casagrande (UNESC)

Prof. Esp. Kristian Madeira (UNESC)

Dedico a meus pais, por acreditar em mim e não medir esforços para ajudar a alcançar meus objetivos, sempre com uma palavra de coragem para não desistir e me guiando pelo caminho certo da vida.

AGRADECIMENTOS

A Deus, por me proporcionar o milagre da vida.

Aos meus pais, Wilson e Edione pela contribuição na formação profissional e pessoal e pela compreensão e apoio que recebi em toda minha vida.

A meu irmão, Claisson pela parceria.

A minha namorada, Luana por me apoiar e me dar força para não deixar desistir nunca dos meus objetivos.

Aos familiares que me ajudaram.

Ao professor Paulo João Martins, pela orientação e paciência durante a realização desse trabalho.

A todos os professores que de alguma forma contribuíram para minha formação.

RESUMO

Devido ao crescimento acelerado da Internet, um grande número de recursos surgiu para acompanhar esse aumento. O conteúdo da Web passou de texto e imagem para multimídia. O uso do tipo de conteúdo exige cada vez mais do servidor, ocasionando uma sobrecarga no tráfego. Este trabalho de pesquisa trata do estudo sobre esse assunto, mais especificamente sobre caracterização da carga de trabalho em um servidor Web. Utiliza para isso um conjunto de oito arquivos de logs de servidores Web Apache reais, que foram utilizados como meio de captura das informações sobre o tráfego. Nesses arquivos em modo texto (bruto), é que ficam gravados todas as requisições HTTP recebidas pelo servidor, e por meio dessas estão dispostas à maioria das informações necessárias para caracterizar a carga. Para as análises, foram utilizadas ferramentas de leitura de logs e geração de cargas sintéticas em servidores, e após isso empregada à metodologia necessária para a caracterização. O estudo concentrou-se na descrição de cada parâmetro que compõe a carga. O levantamento dessas informações poderá ser utilizado para estudos futuros sobre desempenho de servidores Web ou para planejamento de capacidade.

Palavras-chave: Carga de Trabalho; Servidores Web; Otimização.

ABSTRACT

Due to the accelerated growth of the Internet, a great number of resources appeared to accompany that increase. The content of the Web passed of text and image for multimedia. The use of the content type demands more and more from the servant, causing an overload in the traffic. This research work treats of the study on that subject, more specifically about characterization of the work load in a servant Web. It uses for that a group of eight files of logs of servants Web Apache real, which were used as middle of capture of the information on the traffic. In those files in way text (rude), it is that are engravings all of the requests HTTP received by the servant, and through those they are willing to most of the necessary information to characterize the load. For the analyses, tools of logs reading and generation of synthetic loads were used in servants, and after that maid to the necessary methodology for the characterization. The study concentrated on the description of each parameter that composes the load. The rising of that information can be used for future studies on acting of servants Web or for capacity planning.

Keywords: Workload, Web Servers, Optimization.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

ARPA	*Advanced Research Project Agency*
ARPANET	*Advanced Research Projects Agency Network*
ASP	*Active Server Pages*
C.O.V	*Coeficiente de Variação*
CERN	*European Organization for Nuclear Research*
CPU	*Central Processing Unit*
DARPA	*Defense Advanced Research Projects Agency*
DNS	*Domain Name System*
FTP	*File Transfer Protocol*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
I/O	*Input/Output*
IEFT	*Internet Enginnering Task Force*
IP	*Internet Protocol*
JSP	*JavaServer Pages*
MILNET	*Military Network*
MIME	*Multipurpose Internet Mail Extensions*
OLTP	*On-Line Transaction Processing*
OSI	*Open Systems Interconnection*
PHP	*Hypertext Preprocessor*
PSOL	*Probability Statistc Object Library*
RFC	*Request for Comments*
SMTP	*Simple Mail Transfer Protocol*

SPEC	*Standard Performance Evaluation Corporation*
TCP	*Transmission Control Protocol*
TELNET	*Teletype Network*
UDP	*User Datagram Protocol*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locators*
W3C	*World Wide Web Consortium*
WAIS	*Wide Area Information Servers*
WAN	*Wide Area Network*
WWW	*World Wide Web*

SUMÁRIO

1 INTRODUÇÃO

O exponencial crescimento da Web, e um universo de informações e recursos disponibilizados na Internet e o aumento no tráfego, muitas vezes supera a disponibilidade em que os servidores têm em atender uma requisição solicitada pelo cliente, e conseqüentemente aumenta o tempo de resposta.

O desempenho de um sistema depende muito das características de sua carga, assim, o primeiro passo em um estudo de avaliação ou planejamento de capacidade é caracterizar a carga de trabalho. Isso requer a compressão do que está acontecendo em seu ambiente real, que pode ser definida com o conjunto de informações de entrada que o sistema recebe do seu ambiente durante um período de tempo.

Segundo Krishnamurthy e Rexford (2001) entre as principais propriedades que descreve as cargas de trabalho da Web estão: as características das mensagens HTTP, que definem vários métodos de pedidos e uma série de códigos resposta; características do recurso, onde o cliente solicita um recurso com propriedades diversas em termos de tipo de conteúdo, tamanho, popularidade e freqüência. Por fim, o comportamento do usuário e os hábitos de navegação podem produzir um impacto significativo na carga produzida.

Embora cada sistema possa exigir uma técnica específica para a análise e caracterização de sua carga de trabalho, existem algumas orientações gerais que se aplicam bem a todos os sistemas. Segundo Menascé e Almeida (2002), as etapas incluem: A especificação de um ponto de referência a partir do qual a carga será analisada; a escolha de um conjunto de parâmetro que preserva as principais características das cargas; monitoração do sistema para obter os dados brutos; análise e redução de dados; construção do modelo de carga que mantém as informações mais relevantes.

Como é difícil manipular cargas reais, pois possui um tamanho muito grande de elementos, é preciso reduzir as informações, ou seja, criar um modelo, mas que mantenha as características principais das cargas reais. As vantagens em se trabalhar com modelo, é que pode-se alterar os parâmetros da carga para que haja uma reflexão diretamente no sistema, e todas essas implicações podem levar a construção de sistemas melhores.

1.1 OBJETIVO GERAL

Analisar as características e aplicação de metodologias na caracterização da carga de trabalho em um servidor Web em ambiente simulado.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos que compõem essa pesquisa são:

a) identificar as características da carga de trabalho em um servidor Web;
b) estudar uma metodologia para caracterização da carga de trabalho;
c) analisar a carga de trabalho por meio de logs de acesso do servidor;
d) utilizar software para leitura de logs e geradores de sobrecarga de sistema em ambiente simulado;
e) avaliar e analisar as métricas levantadas a partir da medição;

1.3 JUSTIFICATIVA

O objetivo de melhorar o desempenho e muitas vezes saber se o servidor Web está correspondendo em termos de otimização, necessita conhecer os serviços e quais propriedades cada um contém, assim desta forma precisa-se realizar algumas avaliações. Assim, é necessário saber como realizar estas avaliações. Uma forma é conhecer a sua carga de trabalho. A avaliação do desempenho do sistema exige uma forma de analisar a carga estabelecida sobre ele.

A proposta desse trabalho é analisar e descrever as informações necessárias para compreender a carga a que um servidor Web é submetido, com isso utilizando ferramentas

disponíveis para análise de dados de medição, métricas para caracterização de carga de trabalho.

A análise das cargas consiste em captura as principais propriedades do tráfego da Web e representa os resultados de desempenho que ocorrem durante vários períodos do dia. Com isso pode-se analisar quais os tipos de carga que interferem no desempenho de um servidor. Observando que a carga de trabalho não é uma métrica para este trabalho e sim ela é constituída de métricas para realizar as análises. Esta passa a ser uma métrica quando for utilizada para realizar o planejamento de capacidade, mas para este trabalho acaba sendo um método. O desempenho do banco de dados não faz parte da análise, já que os softwares não analisam este tempo, e não há a necessidade de realizar este procedimento, pois este tempo na análise é considerado constante.

O que se deseja é conhecer as métricas de forma a caracterizar o tráfego em um servidor Web, e assim disponibilizar este material como base para outros estudos no futuro.

1.4 ESTRUTURA DO TRABALHO

O desenvolvimento dessa pesquisa trata do tema de caracterização da carga de trabalho em servidores Web, assunto que são muito discutidos em trabalhos correlatos, devido a importância em que se têm em medir qual a carga em que um sistema é submetido e quais afetam mais seu desempenho. Com isso, pode-se utilizar esses levantamentos de métricas para um possível projeto de sistemas melhorados.

Seu desenvolvido foi realizado em 4 capítulos:

O capítulo 1 trata da introdução sobre o tema proposto, junto com os problemas a se resolver assim como os objetivos que se tem em alcançar.

No capítulo 2, são definidos o conceito de Internet e protocolos envolvidos para o funcionamento das redes Web.

Já no capítulo 3, trata-se do assunto mais específico sobre o tema proposto, apontando a metodologia a seguir e características gerais das cargas, que serviu de embasamento para a realização desse trabalho.

O capítulo 4 versa sobre o desenvolvimento do trabalho, em que foi aplicado a metodologia e software para análise e resultados da caracterização da carga de trabalho em um servidor Web.

E por fim as considerações finais sobre os resultados obtidos e sugestões para estudos futuros.

2 CARACTERÍSTICAS DA WEB

A *World Wide Web*, é uma estrutura arquitetônica que permite o acesso a documentos vinculados espalhados por milhões de máquinas na Internet. Em dez anos, ela deixou de ser um meio de distribuição de dados entre pesquisadores físicos para ser tornar a aplicação que milhões de pessoas consideram ser "A *Internet*" (TANENBAUM, 2003).

2.1 EVOLUÇÃO

Sua história começa de certa forma, com a proposta de Vannevar Bush em 1945 para o Memex, descrito como um dispositivo em que um indivíduo armazena todos os seus livros, registros e comunicações, em que é mecanizado de modo que possa ser consultado com velocidade e flexibilidade incríveis. (KRISHNAMURTHY; REXFORD, 2001).

Sua proposta previu a indexação de recursos de multimídias e texto em grande escala, que poderiam ser acessados rapidamente, com isso influenciou pesquisadores nos 50 anos que decorreram. O próximo passo na evolução foi a criação do hipertexto, sendo criado por Ted Nelson, em 1965, para descrever a escrita não seqüencial que apresenta informações como uma coleção de nós interligados, o que exigia vários autores lendo, gravando e revisando o mesmo documento (KRISHNAMURTHY; REXFORD, 2001).

Em 1989 no centro europeu para pesquisa nuclear (CERN), iniciou-se a proposta para uma teia de documentos vinculados partindo do físico Tim Berners-Lee. Dessa proposta nasceu então a *World Wide Web*. O primeiro protótipo em modo de texto, já era operacional um ano e meio depois. Em 1991, foi demonstrado publicamente na conferência *Hypertext* `91. Seu desenvolvimento prosseguiu no ano seguinte com o lançamento da primeira interface gráfica, o Mosaic, em fevereiro de 1993 (TANENBAUM, 1997).

As redes de computadores que é ultilizada hoje têm origem na ARPANET, que foi desenvolvida no final da década de 1960, pela *Advanced Research Projects Agency* (ARPA), agora DARPA (MENASCÉ; ALMEIDA, 2002). Essa rede foi financiada pelo Departamento de Defesa dos Estados Unidos, que estava interessado em estudar técnicas para oferecer comutação de dados independente do fornecedor. Com isso, iniciou-se o esforço para padronizar os protocolos de comunicação da rede de suporte (KRISHNAMURTHY; REXFORD, 2001).

O primeiro protocolo entre servidores foi o *Network Control Protocol* (NCP) que era um protocolo relativamente simples. Na busca por um protocolo que pudesse ser usado em qualquer tipo de rede, se evolui até o protocolo TCP/IP que se tornou o protocolo padrão para interconexão de sistemas abertos (MURHAMMER; GAERTNER, 2000).

A Internet conforme é conhecida hoje, é uma grande coleção de WANs interligadas no mundo inteiro. Os fundamentos da Internet são seus protocolos de rede IP, e de processo, TCP. A mesma experimentou um crescimento exponencial desde o seu início. A quantidade de componentes conectados à Internet cresceu de cerca de 10 nós nos primeiros dias da ARPANET para quase cerca de 100 milhões de nós em menos de 30 anos (MENASCÉ; ALMEIDA, 2002).

Os avanços na tecnologia dos componentes da Internet estão sendo guiados pelas necessidades de novas aplicações. Por isso, é importante ter sempre em mente que é uma infra-estrutura nas quais novas aplicações estão constantemente sendo inventadas e disponibilizadas (KUROSE; ROSS, 2006).

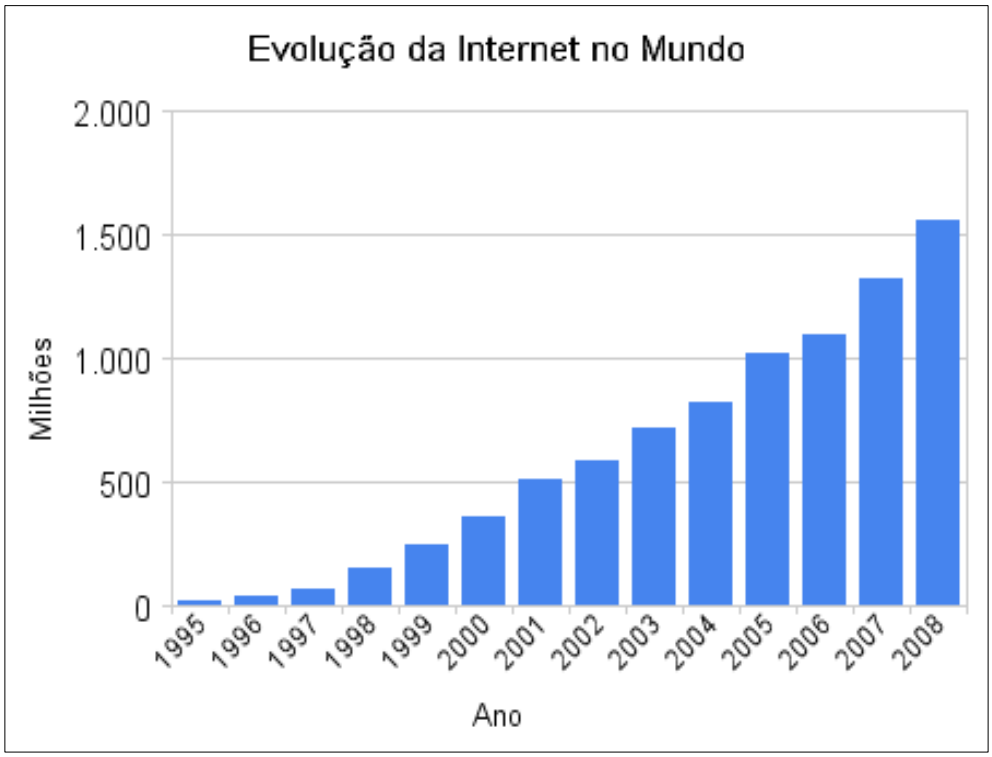

Figura 1. Evolução da Internet no mundo
Fonte: Internet World Stats (2008)

Devido a sua interface gráfica colorida e de fácil utilização ela ganhou uma grande popularidade e com esse crescimento exponencial surgiu à necessidade de desenvolver protocolos para interconexão de sistemas abertos como o modelo OSI e o TCP/IP, esse último o mais usado e padrão para ligação inter-redes. Também houve a necessidade de desenvolver um protocolo que atuasse em uma camada acima do TCP, usado na comunicação entre clientes e servidores Web, então surgiu o protocolo HTTP. Esses protocolos e suas aplicações serão abordados nas próximas seções.

2.2 MODELOS DE REFERÊNCIA

Quando surgiram as redes de computadores, as soluções em tecnologias só eram suportadas por seu fabricante. Não havia a possibilidade de utilizar sistemas de fabricantes

diferentes. Dessa forma um mesmo fabricante era responsável por construir quase tudo na rede.

Para facilitar a interconexão de sistemas de computadores a *International Standards Organization* (ISO) desenvolveu um modelo de referência chamado *Open Systems Interconnection* (OSI), para que os fabricantes pudessem criar protocolos a partir desse modelo.

2.2.1 Modelo de Referência OSI

O modelo de referência OSI, é um modelo de sete camadas de comunicação de dados com transporte físico na camada mais baixa e protocolos de aplicação nas camadas mais altas. Cada camada fornece instruções à camada de cima e em troca conta com as funções fornecidas pela camada de baixo (MURHAMMER; GAERTNER, 2000).

Figura 2. Camadas do modelo de referência OSI
Fonte: MURHAMMER, M; GAERTNER, J. (2000)

2.2.1.1 Camada de Aplicação

Essa camada fornece serviços de rede aos aplicativos do usuário, tais como emulação de terminal e transferência de arquivos (MURHAMMER; GAERTNER, 2000). É diferenciada por não fornecer serviços a nenhuma outra camada, e estabelecer disponibilidade de comunicação (SOARES; LEMOS; COLCHER, 1995).

2.2.1.2 Camada de Apresentação

Certifica-se que a informação emitida pela camada de aplicação seja legível para outro sistema. Se necessário faz a conversão de vários formatos de dados usando um formato comum (SOARES; LEMOS; COLCHER, 1995). Em outras palavras faz a formação de dados e criptografia (MURHAMMER; GAERTNER, 2000).

2.2.1.3 Camada de Sessão

Permite que dois programas em computadores diferentes estabeleçam uma sessão de comunicação, ou seja, gerência e fornece seus serviços à camada de apresentação que sincroniza a conversação entre as camadas de apresentação dos dois *hosts* fazendo o controle da troca de dados. (SOARES; LEMOS; COLCHER, 1995).

2.2.1.4 Camada de Transporte

Esta desmonta os dados do sistema *host* que está enviando e monta-os novamente em uma seqüência no que está recebendo. Enquanto as camadas de aplicação, de apresentação e de

sessão estão relacionadas a problemas de aplicativos, as quatro inferiores estão relacionadas a problemas de transporte de dados. Esta camada fornece um serviço de transporte de dados, oferencendo serviços de comunicação, que mantém e termina corretamente o caminho entre o transmissor (origem) e o receptor (destino), oferecendo serviço confiável que usa o controle do fluxo de informações e a detecção e recuperação de erros (SOARES; LEMOS; COLCHER, 1995).

2.2.1.5 Camada de Rede

Fornece conectividade entre dois sistemas *hosts* que podem estar localizados em redes separadas (SOARES; LEMOS; COLCHER, 1995). Pode-se dizer que essa faz a entrega de pacotes, incluindo roteamento (MURHAMMER; GAERTNER, 2000).

2.2.1.6 Camada de Enlace de Dados

Especifica como organizar dados em quadros e como transmiti-los por meio de uma rede (COMER, 2001). Fornecem dados por meio de um link físico, com isso, trata do endereçamento físico, da topologia, do acesso de rede e do controle de fluxo (SOARES; LEMOS; COLCHER, 1995).

2.2.1.7 Camada Física

Corresponde ao hardware de rede básico (COMER, 2001). Define as especificações mecânicas, funcionais, elétricas e de procedimentos para ativar, manter e desativar o link físico entre sistemas finais (SOARES; LEMOS; COLCHER, 1995).

Os protocolos OSI se desenvolveram vagarosamente, e como executar uma pilha completa desses, exigem muitos recursos, eles não foram amplamente desenvolvidos, especialmente no mercado de computadores de mesa e de pequeno porte. Enquanto isso, o TCP/IP e a Internet estavam se desenvolvendo e sendo colocados em uso rapidamente (MURHAMMER; GAERTNER, 2000).

2.3 MODELO DE REFERÊNCIA TCP/IP

Foi o primeiro conjunto de protocolos desenvolvido para o uso em uma inter-rede (COMER, 2001). É a sigla dada para *Transmission Control Protocol/Internet Protocol* surgiu com a Internet, como solução para a interligação dos computadores nessa rede. (MATTHEW, 1997). Esse é o padrão mundial para interconexão de sistemas abertos e é executado em mais tecnologia de rede do que qualquer outro conjunto de protocolos (COMER, 1999).

2.3.1 Protocolos TCP/IP

Os Protocolos TCP/IP além das conexões à Internet, também são empregados por muitas organizações em suas redes internas. Uma rede privativa que os utiliza é conhecida como uma *intranet.* Empresas dos setores aeroespacial, automotivo, eletrônico entre outras as utilizam para conectar todas as suas redes remotas e locais (COMER, 1999).

Segundo Comer e Hunt (2002) o protocolo TCP/IP aborda algumas das principais características que os diferencia dos demais:

a) uso de padrões abertos**:** está disponível na *Web* por meio das *Request for Comments* (RFCs), padronizada pela *Internet Enginnering Task Force* (IETF);

b) independência da tecnologia de rede: a comunicação é feita por meio da comutação de pacotes entre as partes e não utiliza nenhuma tecnologia específica, tanto de hardware como de software. Opera sobre uma variedade de protocolos da camada física e de enlace de dados;
c) protocolos padronizados: estes padrões facilitam a criação de novas aplicações envolvendo a rede. As modificações em um nível não afetam outros;
d) interconexão total: o *host* recebe um endereço único permitindo o acesso de qualquer parte da rede, definindo sua identificação;
e) confirmações fim-a-fim: possui um mecanismo de reconhecimento permitindo a origem identificar se uma mensagem foi recebida corretamente.

2.3.2 Arquitetura TCP/IP

Essa se baseia principalmente em serviço de transporte orientado á conexão, fornecido pelo TCP, e serviço de rede não-orientado à conexão, fornecido pelo IP.

A arquitetura da Internet dá uma ênfase toda especial a interligação de diferentes tecnologias de redes. Portanto, a única forma de permitir que um grande volume de usuários possa trocar informações é interligá-las às quais eles estão conectados, formando assim uma *inter-rede* (SOARES; LEMOS; COLCHER, 1995).

Segundo Murhammer e Gaertner (2000) os objetivos do projeto do TCP/IP incluiam dois fundamentos:

a) construir uma interconexão de rede que fornecesse serviços de comunicação universal. Cada rede física possui sua própria interface de comunicação dependente de tecnologia;

b) interconectar redes físicas diferentes para formar o que aparece ao usuário como uma grande rede, chamado de *internetwork* ou Internet.

Na interligação de duas redes distintas é necessário conectar uma máquina a ambas as redes. Esta fica responsável pela tarefa de transferir mensagens de uma rede para a outra. Uma máquina que conecta duas ou mais redes é denominada *Internet gateway* ou *Internet router*. Para ser capaz de rotear corretamente mensagens, os *gateways* precisam conhecer a topologia das inter-redes, ou seja, precisam saber como estão interconectadas (SOARES; LEMOS; COLCHER, 1995).

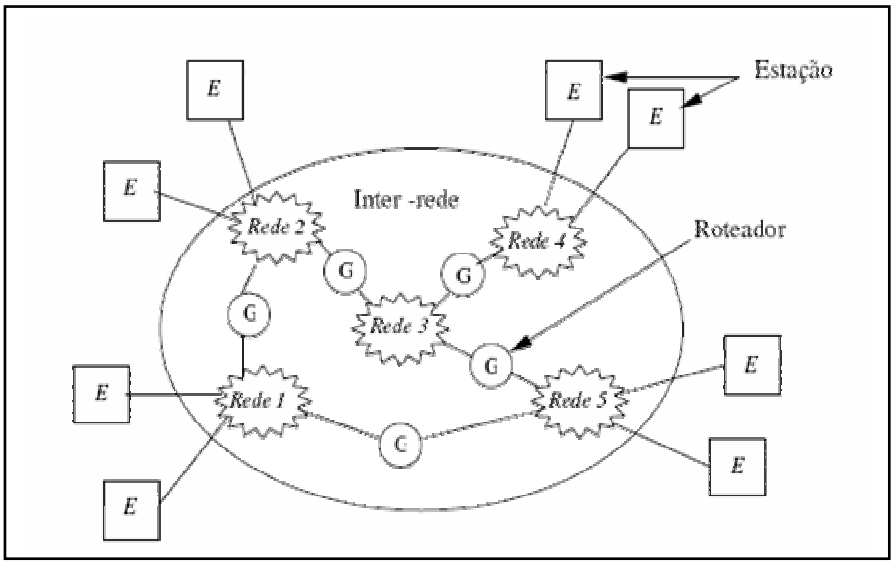

Figura 3. Conceito de inter-rede
Fonte: SOARES, L; LEMOS, G; COLCHER, S. (1995)

A arquitetura Internet TCP/IP é organizada em quatro camadas ou níveis conceituais construídas sobre uma quinta camada que não faz parte do modelo, a camada intra-rede.

Figura 4. Camadas da arquitetura TCP/IP
Fonte: MURHAMMER, M; GAERTNER, J. (2000)

2.3.2.1 Camada de Aplicação

Utilizam programas de aplicação para acessar os serviços disponíveis na inter-rede. Essas interagem com o nível de transporte para enviar e receber dados, e podem usar o serviço orientado à conexão, fornecido pelo TCP ou serviço não orientado à conexão, fornecido pelo *User Datagram Protocol* (UDP).

Algumas aplicações segundo Soares, Lemos e Colcher (1995), disponíveis na internet TCP/IP são:

a) *Simple Mail Transfer Protocol* (SMTP), que oferece um serviço store-and-forward para mensagens que carregam correspondências contendo textos;

b) *File Transfer Protocol* (FTP), que fornece o serviço de transferência de arquivos;

c) TELNET, que oferece o serviço terminal virtual, permitindo que o usuário de um computador estabeleça *login* em uma máquina remota;

d) *Domain Name System* (DNS) que oferece um serviço de mapeamento de nomes em endereços de rede;

e) *HiperText Transfer Protocol* (HTTP*),* usado para buscar páginas na *World Wide Web* (WWW).

2.3.2.2 Camada de Transporte

Esta tem a finalidade de permitir que a entidade par dos hosts de origem e de destino mantenham uma conversação. Dois protocolos foram definidos aqui: O TCP, que é orientado á conexão confiável que permite a entrega sem erros de um fluxo de bytes originado de uma determinada máquina em qualquer computador da inter-rede. O segundo protocolo dessa camada, o UDP, é um protocolo sem conexão não confiável para aplicações que não necessitam de controle de fluxo, e nem de manutenção da seqüência das mensagens enviadas. Ele é amplamente usado em aplicações em que à entrega imediata é mais importante do que ela precisa como a transmissão de dados de voz ou de vídeo (TANENBAUM, 1997).

2.3.2.3 Camada de Inter-Rede

Responsável pela transferência de dados por meio da inter-rede, desde a máquina de origem até a de destino (SOARES; LEMOS; COLCHER, 1995). Essa camada integra toda a arquitetura, sua tarefa é permitir que os hosts injetem pacotes em qualquer rede e garantir que eles sejam transmitidos independentemente do destino (TANENBAUM, 1997).

2.3.2.4 Camada de Interface de Rede

A arquitetura TCP/IP não faz nenhuma restrição às redes que são interligadas para formar a inter-rede. Portanto, qualquer tipo pode ser ligada, bastando para isso que seja

desenvolvida uma interface que compatibilize a tecnologia especifica da rede com o protocolo IP (SOARES; LEMOS; COLCHER, 1995).

2.3.2.5 Camada Intra-Rede

Corresponde ao hardware de rede básico da mesma maneira que a camada 1 no modelo de referência OSI de sete camadas (COMER, 2001).

2.4 COMPARAÇÃO ENTRE OS MODELOS DE REFERÊNCIA

Os modelos de referência OSI e TCP/IP têm muito em comum. Esses baseiam no conceito de uma pilha de protocolos independentes. Além disso, as camadas têm praticamente as mesmas funções, apesar dessas semelhanças fundamentais, os dois modelos também têm muitas diferenças (TANENBAUM, 1997).

A primeira diferença entre as arquiteturas está no número e camadas. Enquanto na arquitetura OSI são definidas sete camadas, na arquitetura TCP/IP são definidas cinco (SOARES; LEMOS; COLCHER, 1995).

O modelo OSI tem como conceitos fundamentais serviços, interfaces e protocolos. Provavelmente a maior contribuição é tornar explícita a distinção entre esses três conceitos. Cada camada executa alguns serviços para a camada acima dela. Os protocolos utilizados em uma camada são de responsabilidade dessa mesma (TANENBAUM, 1997).

Originalmente, o modelo TCP/IP não distinguiu com clareza a diferença entre serviço, interface e protocolo, embora as pessoas tenham tentado adaptá-las ao modelo OSI. O modelo de referência OSI foi concebido *antes* de os protocolos terem sido inventados. Conseqüentemente, o modelo não foi criado com base em um determinado conjunto de

protocolos, o que o deixou bastante flexível. Já com o TCP/IP, aconteceu exatamente o contrário, como os protocolos vieram primeiro, o modelo foi criado com base neles, e não tiveram problemas ao se adaptar (TANENBAUM, 1997).

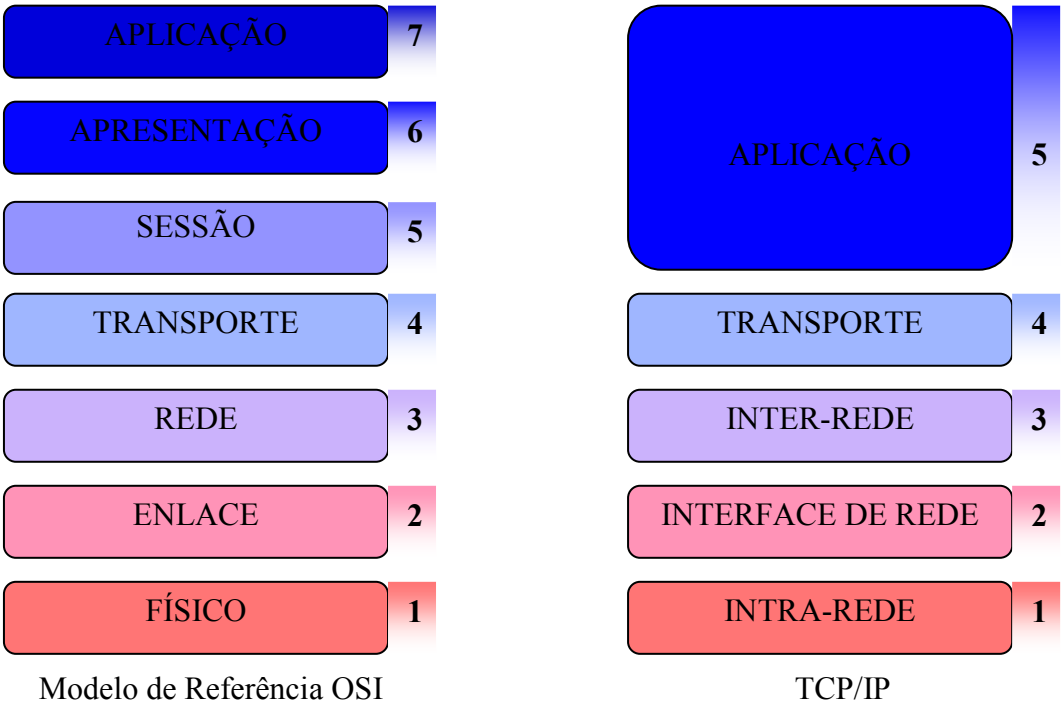

Figura 5. Comparação com o modelo de referência OSI e TCP/IP
Fonte: Adaptado de MURHAMMER, M; GAERTNER, J. (2000)

2.5 PROTOCOLO HTTP

O *HiperText Transfer Protocol* é um protocolo em nível de aplicação, disposto em cima do TCP, usado na comunicação entre clientes e servidores (MENASCÉ; ALMEIDA, 2002).

A operação da *Web* depende de se ter um modo padronizado e bem definido para os componentes se comunicarem. O HTTP é o protocolo de pedido e resposta que oferece suporte para *World Wide Web*, é o modo mais comum de se transferir recursos. O mesmo define o formato e o significado das mensagens trocadas entre os componentes da Web, como clientes e servidores (KRISHNAMURTHY; REXFORD, 2001).

Apesar da habilidade em negociar formatos, o WWW possui uma linguagem básica de intercâmbio de hipertexto, denominada *HyperText Markup Language* (HTML) (SOARES; LEMOS; COLCHER, 1995). Essa permite descrever a estrutura dos documentos, indicando títulos, ênfase, links para outros documentos, além de poder estar contido imagens e outros objetos de multimídia (MENASCÉ; ALMEIDA, 2002).

O HTTP pode ser usado para fazer acesso a documentos em conjunto não limitados e extensíveis de formatos. Para que isso seja possível, o cliente ao solicitar a transferência, de uma cópia de um objeto, envia uma lista com os formatos que pode manipular. O servidor responde enviando o objeto solicitado codificado em um dos formatos informados pelo cliente (SOARES; LEMOS; COLCHER, 1995).

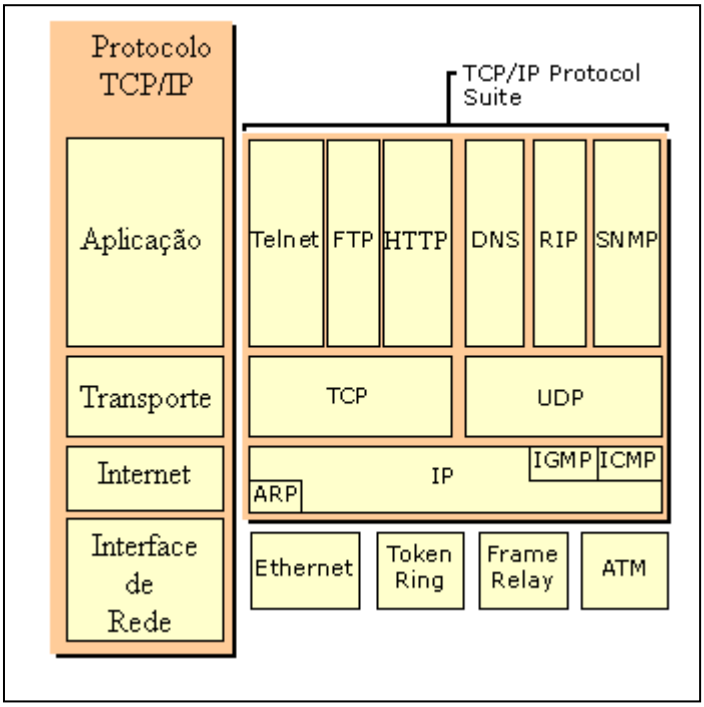

Figura 6. Arquitetura do protocolo TCP/IP
Fonte: KUROSE, J; ROSS, K. (2006)

2.5.1 Mensagem de Requisição e Resposta

Uma Mensagem HTTP é uma seqüência de octetos enviada por uma conexão de transporte, pode ser um pedido enviado de um cliente para um servidor ou uma resposta enviada de um servidor para um cliente (KRISHNAMURTHY; REXFORD, 2001).

Ele possui dois tipos de mensagem, as requisições e as respostas. As requisições consistem em uma linha de informação a ser executada no servidor, seguida de uma ou mais linhas de cabeçalho contento parâmetros e corpo da mensagem, como descrito por Coulouris et al (2001), a seguir:

a) *linha de requisição*: possui o método HTTP invocado, a localização do objeto no servidor e a versão do protocolo utilizada;

b) *linhas de cabeçalho*: o cabeçalho contém campos que permitem ao cliente utilizar para envio de informação para o servidor;

c) *corpo da mensagem*: este possui dados adicionais para serem entregues ao servidor, esse campo às vezes são usados o mesmo ocorre com o método POST.

A Figura 7 descreve melhor o formato geral de uma mensagem de requisição HTTP. Sendo os campos cr (carriage return) e lf (line feed).

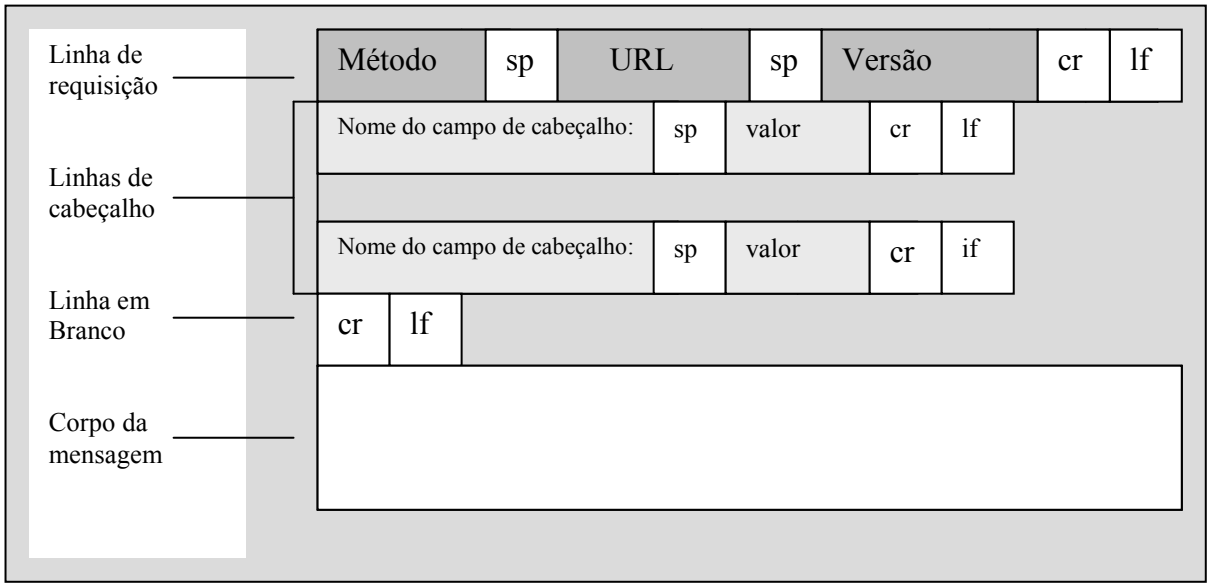

Figura 7. Formato geral de uma mensagem de requisição
Fonte: KUROSE, J; ROSS, K. (2006)

Na mensagem de requisição, define-se um conjunto de métodos para indicar seu propósito. Esses métodos funcionam como comandos enviados ao servidor (SILVA, 2006). A Tabela 1 descreve os métodos de acesso utilizado pelo cliente (*browser*) para invocar um pedido de um objeto ao servidor.

Tabela 1. Métodos de Requisição do Protocolo HTTP /1.1

Métodos	Descrição
GET	Solicita que seja retornado o recurso identificado pela URL.
HEAD	Obtém informação sobre o recurso sem que o mesmo seja retornado ao cliente. Checa a validade de links, acessibilidade e a data da última modificação.
POST	Envia informações adicionais do cliente para o servidor no corpo da mensagem.
PUT	Permite criar e modificar um recurso no servidor web.
DELETE	Solicita que o servidor web remova o objeto definido pela URL.
OPTIONS	Permite determinar as opções e requisitos associados aos recursos solicitados, sem necessariamente iniciar sua recuperação.
TRACE	Usado para enviar uma mensagem de teste ao servidor do tipo *loopback.*
CONNECT	Reservado para comunicação do servidor *Proxy*.

Fonte: Coulouris, G. et al (2001)

Segundo Coulouris et al (2001), as mensagens de resposta seguem como descrito abaixo:

a) *linha de estado*: armazena o código de resposta (status) e versão do protocolo;
b) *linhas de cabeçalho*: vários campos que informam as características do servidor e do objeto retornado para o cliente;

c) corpo da mensagem: contém informações sobre o objeto retornado.

Assim como na mensagem de requisição a Figura 8 ilustra o formato geral de uma mensagem de resposta HTTP.

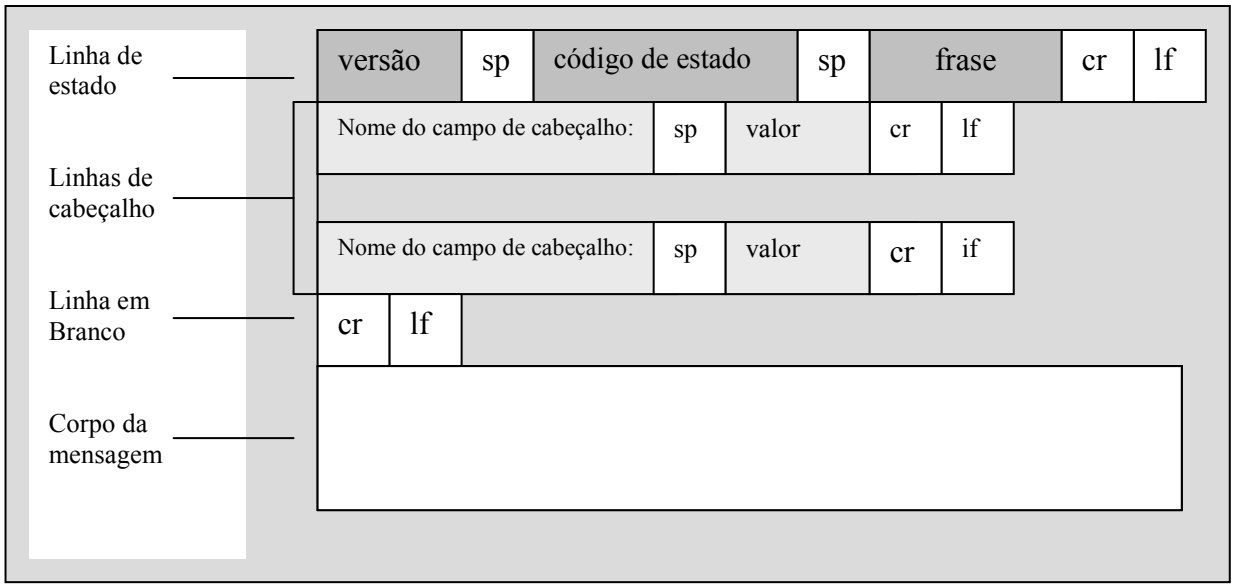

Figura 8. Formato geral mensagem de resposta
Fonte: KUROSE, J; ROSS, K. (2006)

O protocolo HTTP oferece uma lista de códigos de *status* para o servidor web informarem ao cliente o grau de sucesso ao tentar atender a requisição. A tabela 2 mostra as classes de códigos de resposta.

Tabela 2. Classes de código de resposta do protocolo HTTP/1.1

Classes	Descrição
1xx	Requisição recebida em fase de processamento.
2xx	Requisição recebida, entendida e processada com sucesso.
3xx	Redireciona o cliente para outra URL.
4xx	Requisição Inválida.
5xx	Incapacidade do servidor em processar a requisição.

Fonte: COULOURIS et al. (2001)

2.5.2 Evolução do HTTP

O protocolo HTTP evoluiu juntamente com a Web e seus dois outros componentes, o URI[1] e o HTML. Segundo Krishnamurthy e Rexford (2001) esse foi em duas fases distintas:

a) da primeira especificação HTTP/0.9 até a versão HTTP/1.0, por um período de quadro anos;

b) do HTTP/1.0 até o HTTP/1.1 em outros quatro anos.

Esse foi proposto por Tim Berners-Lee em março de 1990, nos laboratórios do CERN[2], como um mecanismo suficiente poderoso para acessar documentos em qualquer lugar da Internet e para ajudar a navegar entre eles por meio de *links* de *hipertexto*. A versão

[1] Uniform Resource Identifiers (URI) O mecanismo de nomeação permite que os recursos residam em qualquer lugar na Internet e separa noção de um recurso e a de uma resposta.
[2] CERN, o laboratório Europeu de Física da Partícula, próximo a Genebra .

mais antiga do protocolo, chamada HTTP/0.9, foi descrita inicialmente em janeiro de 1992. A especificação do mesmo levou à codificação das regras da interação entre clientes e servidores.

Em dezembro do mesmo ano, surgiu uma discussão na lista de correspondência www-talk a respeito do potencial para se criar sistemas de recuperação de informação como *Wide Area Information Servers* (WAIS) e o compatível da *Web* com MIME[3]. As respostas HTTP poderiam ser retornadas em mensagens MIME estendidas. A primeira especificação de rascunho da versão 1.0 foi feita em março de 1993. Em junho do mesmo ano, a especificação de rascunho do *HyperText Markup Language* propôs a esta como sendo um tipo de conteúdo MIME. Mais adiante nesse ano, foi publicada a primeira especificação *Draft* da Internet para os *Uniform Resource Locators* (URL) após discussões dentro do grupo de trabalho do URI (KRISHNAMURTHY; REXFORD, 2001).

Alguns rascunhos dos documentos versão 1.0 vieram em seguida, em maio de 1996, para capturar o estado das implementações do HTTP, foi lançado um *Informational Request For Comments*, RFC 1945. Isso serviu como base para a maioria das implementações de componentes dessa versão. Os três anos seguintes vieram muitas atividades para atualizar o HTTP/1.0 a fim de resolver uma série de problemas (KRISHNAMURTHY; REXFORD, 2001).

Em 1998, a *World Wide Web Consortium* (W3C) criou o HTTP/1.1, conhecido como HTTP-NG (HTTP *New Generation*), uma versão compatível com as anteriores e com os modos mais eficazes. Essa usa um esquema de conexões persistentes, o que permite várias transações HTTP em utilizar a mesma conexão TCP, aumentado a eficiência da comunicação e evitando a fase de estabelecimento e término de conexões intermediárias, *o three-way handshake.* Ele também possui uma característica conhecida como *pipelining*, neste caso,

[3] MIME era um padrão da Internet em desenvolvimento na época, classificava formatos de dados e aceitava mensagens em várias partes a fim de enviar documentos em várias partes por e-mail.

várias requisições são enviadas em seqüência dentro do mesmo fluxo, sem aguardar pelas respostas. Os novos comandos de *cachê* introduzido também reduzem o tráfego da rede e permite ao protocolo uma maior eficiência (SILVA, 2006).

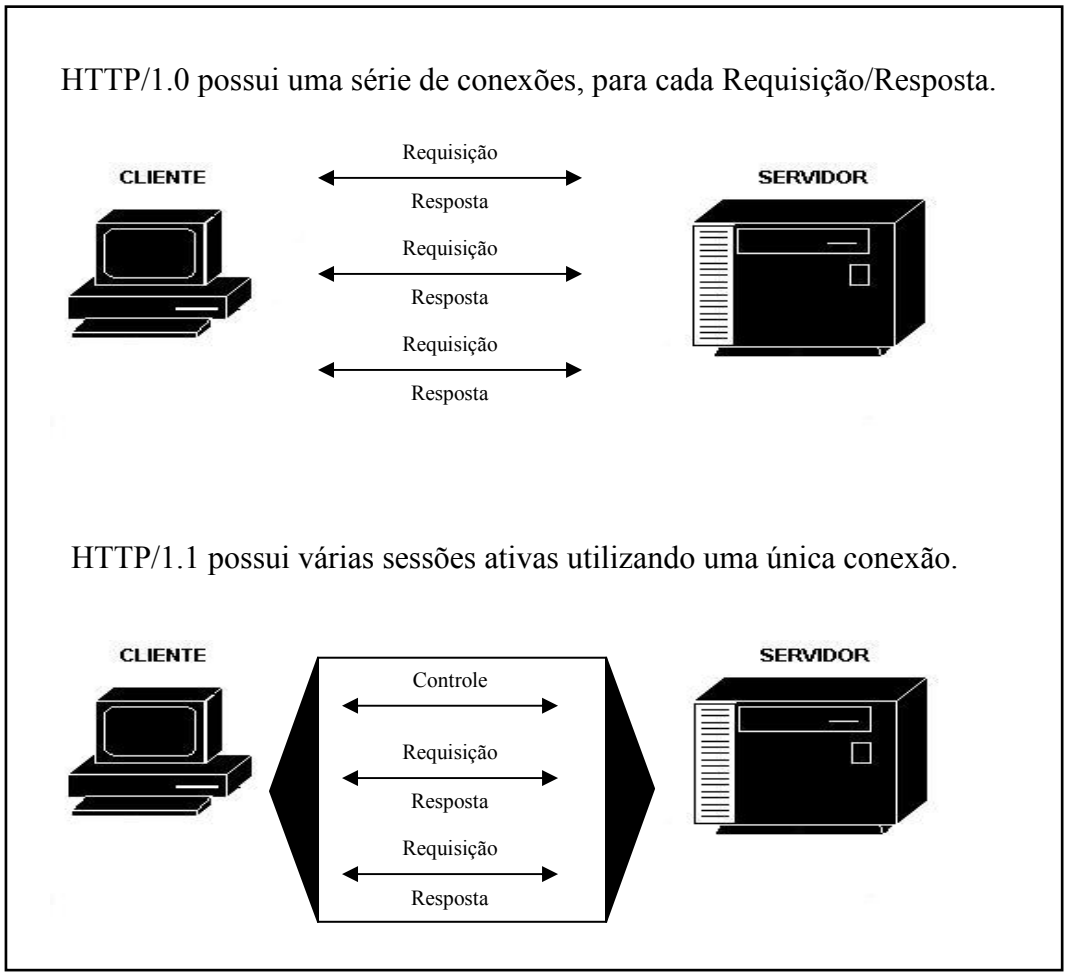

Figura 9. Funcionamento do protocolo HTTP/1.0 e HTTP/1.1
Fonte: Adaptado de SILVA, L., (2001)

Na próxima sessão será discutida sobre servidores Web, sua arquitetura, assim como tratamento de pedido do cliente.

2.6 SERVIDOR WEB

Esse é um programa que trata de pedidos HTTP por recursos específicos (KRISHNAMURTHY; REXFORD, 2001). É uma combinação de uma plataforma de hardware, sistema operacional, software servidor e conteúdo (MESNACÉ; ALMEIDA, 2002).

O servidor Web é executado em uma plataforma de suporte que consiste em um computador com acesso à rede. Esse pode ser disponibilizado com um ou mais processadores e memória e talvez um disco para armazenar documentos estáticos e scripts. As plataformas variam bastante em termos do seu poder de computação, conectividade de rede e capacidade de armazenamento, dependendo do tipo de recursos da *Web* e da taxa esperada de pedidos do cliente. Esse também interage com o processador, disco e conexão de rede por meio do sistema operacional de suporte. Como parte do atendimento a um pedido, o servidor pode invocar um *script* que interage com outros servidores. As interações com outros servidores podem envolver outros protocolos além do HTTP (KRISHNAMURTHY; REXFORD, 2001).

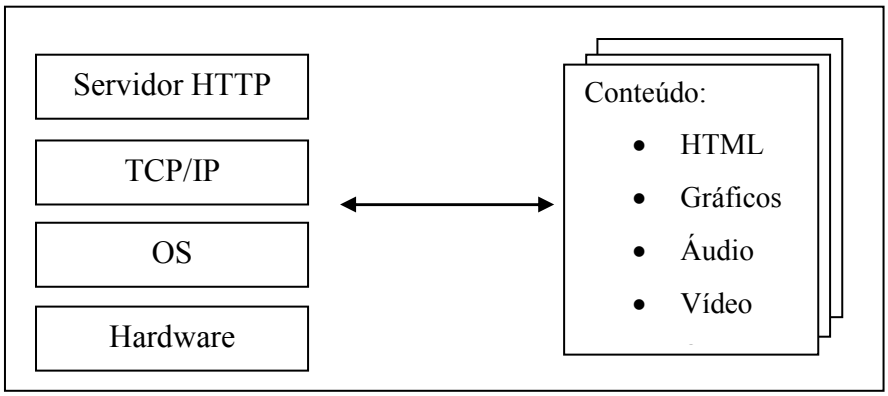

Figura 10. Elementos do servidor Web
Fonte: MESCANÉ, D; ALMEIDA, V. (2002)

2.6.1 Arquitetura de Servidor Web

Um servidor da Web lida com vários pedidos de clientes ao mesmo tempo. Estes pedidos precisam compartilhar o acesso ao processador, disco, memória e interface de rede no servidor. Para tratar os diferentes pedidos dos clientes, pode ter arquitetura tais como:

a) *controlado por eventos,* onde possui um único processo que altera entre atendimentos a diferentes pedidos;

b) *controlado por processo,* aloca cada pedido a um processo separado.

2.6.1.1 Servidores Controlados por Evento

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).

2.6.1.2 Servidores Controlados por Processo

Existe um processo mestre que fica escutando as novas conexões de clientes, este se divide 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* 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).

2.7 QUALIDADE DE SERVIÇO NA WEB

O *Quality of Service* (Qos), surgiu com finalidade de suprir as novas necessidades de comunicação e resolver alguns problemas que impedem o desenvolvimento de algumas aplicações.

Pensando em qualidade, é possível oferecer maior garantia e segurança nas aplicações para Internet, uma vez que o tráfego de aplicações avançadas como voz sobre IP e vídeo-conferência passam a ter maior prioridade, enquanto usuários de aplicações tradicionais continuam utilizando o melhor esforço.

A arquitetura do Qos é dividida em duas implementações distintas: serviços integrados e serviços diferenciados. Os integrados é um modelo baseado em reserva de recursos, enquanto os diferenciados é uma proposta onde os pacotes são marcados de acordo com classes de serviços pré-determinadas (STARDUST, 1999).

2.7.1 Serviços Integrados

Nos Serviços Integrados (*Intserv*) o emissor solicita ao receptor a alocação dos recursos necessários para definir-se uma boa qualidade na transmissão dos dados. O protocolo *Resource Reservation Protocol* (RSVP) é utilizado para troca de mensagens de controle de alocação dos recursos. A alocação desses recursos é dada pela largura de banda e ao tempo

em que será mantida a conexão. O emissor desse serviço tem uma faixa da largura de banda disponível para transmitir seus dados (KUROSE; ROSS, 2006).

2.7.2 Serviços Diferenciados

A arquitetura de serviços diferenciados (*Diffserv)* visa a ministrar diferenciação de serviços escalável e flexível, ou seja, tem a capacidade de manipular diferentes classes de tráfego de diversas maneiras dentro da internet. A necessidade de escalabilidade surge do fato de que milhares de fluxos de tráfego fonte-destino simultâneos podem estar presentes em um roteador de backbone da Internet. E a flexibilidade surge do fato de que novas classes de serviços podem surgir e as velhas podem se tornar obsoletas.

Segundo Kurosse e Ross (2006) a arquitetura Diffserv consiste em dois conjuntos de elementos funcionais:

a) *classificação de pacotes e condicionamento de tráfego*: Na borda de entrada da rede os pacotes que chegam são marcados. Mais especificamente, o campo *Differentiated Service* (DS) do cabeçalho do pacote é configurado para algum valor. A marca que um pacote recebe identifica a classe de tráfego à qual pertence, assim diferentes classes de tráfego receberão serviços diferenciados dentro do núcleo da rede.
b) *função central*: Quando um pacote marcado com DS chega a um roteador habilitado para *Diffeserv*, ele é repassado até seu próximo salto de acordo com o comportamento por salto associado à classe do pacote. Esse comportamento influencia a maneira pela qual os *buffers* e a largura da banda de um roteador são compartilhados entre as classes de tráfego concorrentes.

Os *Diffserv* tem sido o modelo mais utilizado para implementação de QoS. Ele exige menos dos roteadores, precisa de pouca atualização de software para fornecer bons métodos de classificação, policiamento, montagem e remarcação de pacotes.

Neste capítulo foi apresentada a estrutura da Web assim como os protocolos envolvidos para o funcionamento de redes da Internet. O próximo capítulo irá apresentar o conceito de carga de trabalho assim como a metodologia usada para melhor descrevê-las, os principais parâmetros que envolvem a caracterização.

3 CARACTERIZAÇÃO DA CARGA DE TRABALHO

Medições de desempenho são importantes, tais como latência, desempenho de um servidor web entre outras. Estão relacionados diretamente com as características de sua carga, assim primeiramente deve-se entender e caracterizar a carga de trabalho (KRISHNAMURTHY; REXFORD, 2001).

A carga de trabalho de um sistema pode ser definida como o conjunto de todas as informações de entrada que o sistema recebe do seu ambiente durante qualquer período de tempo determinado (MENASCÉ; ALMEIDA, 2002).

Avaliar a carga é muito importante para identificar como um servidor da Web está desempenhando seu papel em relação as diferentes solicitações e respectivamente as diferentes cargas que está sendo submetido, avaliando assim, se está atendendo as expectativas dos clientes que dispõe de seus serviços.

Para descrevê-la é preciso reduzir e resumir as informações, sendo necessária à criação de um modelo que mantenha as características mais relevantes do ambiente. O mesmo leva vantagens quando comparado com as cargas reais, sendo possível alterar os parâmetros do modelo para que haja mudanças no sistema ou na carga. Pode-se então aumentar e modelar apenas mudando um parâmetro, seja o tempo de entrada ou o de processamento, assim os modelos facilitarão o modo de lidar com grandes números de elementos, podendo levar a melhoria de projetos de sistemas (MENASCÉ; ALMEIDA, 2002).

Um processo de caracterização gera um modelo o qual é usado em diversas tarefas, que gera parâmetros com características capazes de dirigir os modelos de desempenho usados para atividades de planejamento de capacidade.

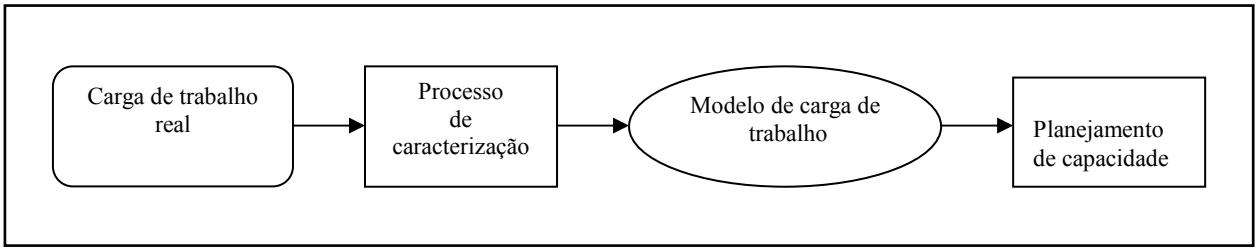

Figura 11. Processo de caracterização da carga de trabalho
Fonte: MENASCÉ, D; ALMEIDA, V. (2002)

A carga de trabalho de um sistema de computador pode ser descrita em diferentes níveis, no mais alto a caracterização comercial descreve os planos corporativos e comerciais úteis para caracterizar em termos de recursos das principais aplicações. Já a caracterização funcional descreve os programas, requisições ou aplicações que compõe a carga, como pode-se observar na Figura 12, que consiste em várias requisições Web, esse tipo de caracterização está relacionado aos serviços de software (por exemplo, *On-Line Transaction Processing* (OLTP), e serviços HTTP) usados para processar a carga de trabalho.

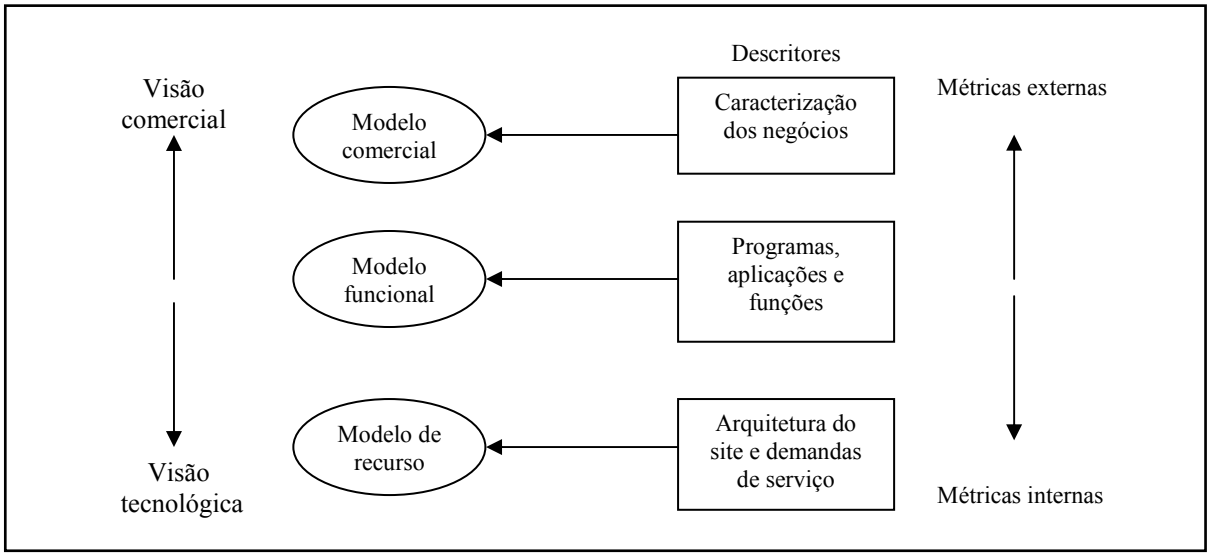

Figura 12. Níveis de descrição da carga de trabalho
Fonte: MENASCÉ, D; ALMEIDA, V. (2002)

Essas atividades normalmente exigem informações quantitativas, na qual uma requisição há um grande documento gráfico consome muito mais CPU, tempo de I/O e

largura de banda da rede do que uma requisição para um pequeno documento HTML, assim, para tornar a caracterização útil, é necessário incluir no modelo, informações sobre requisitos do recurso, os seja, incluir o tamanho do documento (MENASCÉ; ALMEIDA, 2002).

Para obter o resultado esperado, seja no desempenho ou na capacidade de um servidor Web, aplica-se aos modelos da carga de trabalho, que utilizam números menores de parâmetros tornando-se mais fáceis de manipular os dados, sempre respeitando as características das cargas reais, onde avalia-se o servidor quando submetido a uma carga de trabalho real, poderá ter uma melhoria no seu desempenho ou na capacidade.

3.1 METODOLOGIA PARA CARACTERIZAÇÃO DA CARGA DE TRABALHO

Para a criação de um modelo de carga, utilizou-se de uma metodologia que melhor descreve as informações que devem compor esses aspectos.

3.1.1 Definição da Perspectiva de Análise

Do ponto de vista do servidor, a carga de trabalho é formada por todos os tipos de requisições que este recebe de todos os seus possíveis clientes.

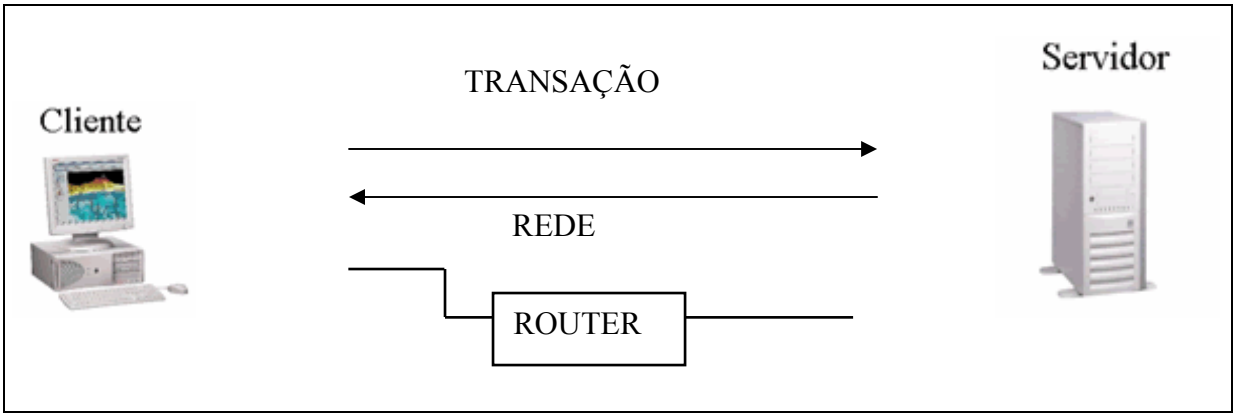

Figura 13. Transação entre cliente e servidor
Fonte: MENASCÉ, D; ALMEIDA, V. (1998)

3.1.2 Identificação dos Componentes Básicos

Transações e requisições são mais comuns para identificar os componentes que compõem a carga, a escolha dos componentes depende da natureza do sistema e do propósito da caracterização.

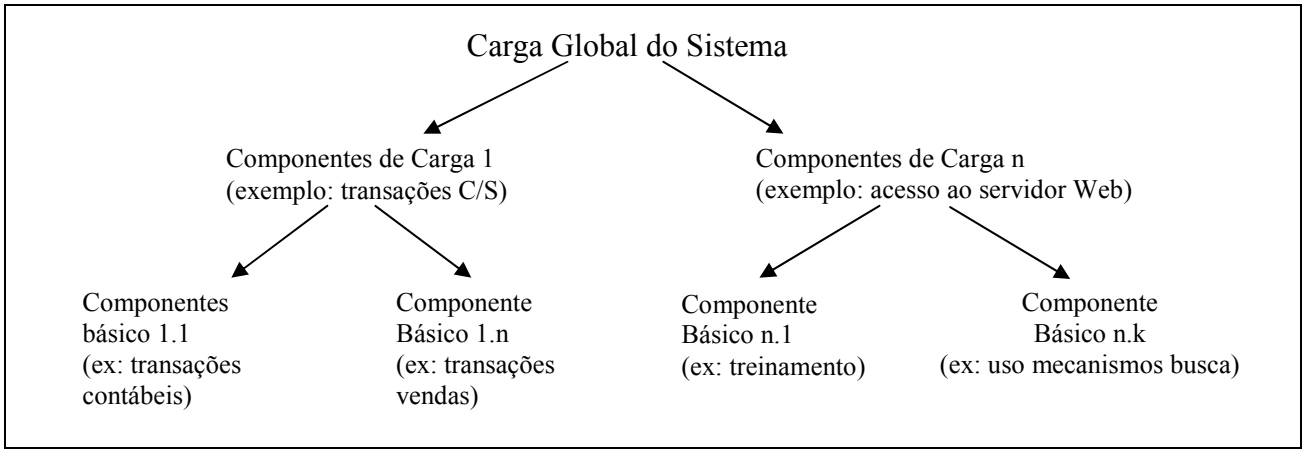

Figura 14. Componentes básicos da carga de trabalho
Fonte: MENASCÉ, D; ALMEIDA, V. (1998)

3.1.3 Definição de Parâmetros que Melhor Caracterizem a Carga

Depois de identificados os componentes básicos da carga de trabalho, é preciso escolher quais parâmetros melhor caracterizam cada um deles, qual a característica comum destes elementos são as exigências dos parâmetros de entrada em modelos analíticos ou de simulação. Segundo Menascé e Almeida (1998) dividide-se em:

a) intensidade: taxa de chegada, número de clientes, número de processo ou threads em execução simultânea;

b) demanda: uma requisição (componente básico) demanda 0,0095 seg; de CPU e 0,04 seg. de I/O no servidor (recurso).

3.1.4 Monitoração e Coleta de Dados

É nesta etapa que inicia a determinação dos valores que serão aos parâmetros de cada componente do modelo. Segundo Menascé e Almeida (1998) as seguintes tarefas devem ser realizadas:

a) identificar a janela de tempo que define a sessão de medição que corresponde a um intervalo, no qual o sistema, a carga e os índices de desempenho serão observados;

b) monitorar e medir as atividades do sistema durante o intervalo definido (janela de tempo), a isso é atribuído o uso de ferramentas disponíveis no SO ou monitores de software que podem ser empregados;

c) a partir dos dados, atribuir os valores necessários aos parâmetros de cada componente.

Muitas vezes, não é possível conseguir dados de demanda por serviços diretamente obtidos deles, neste caso é preciso realizar relacionamentos e cruzamentos de dados para alcançar o objetivo, sendo assim, deve-se assumir os interesses pela demanda de serviços de requisições HTTP a arquivos de imagens em um disco de um servidor Web, onde uma análise do LOG, revelará quantas imagens e seus tamanhos que foram lidas durante o intervalo de medição. Portanto estas informações, combinadas com as características de desempenho do disco, podem ser empregadas para obter demanda pelo serviço desejado.

3.1.5 Partição da Carga de Trabalho

Cargas de trabalho reais podem ser entendidas como uma coleção de componentes heterogêneos, considerando o nível de utilização de recursos. Uma requisição de um vídeo-

clip, difere de uma requisição de um pequeno documento HTML, por isso, representar a carga por uma única classe pode deixar muito a desejar em termos de precisão de resultados.

Particionar a carga implica em dividi-la numa série de classes, de tal forma que sua população seja formada por elementos quase homogêneos, sendo assim o principal problema aqui é definir a semelhança. Segundo Menascé e Almeida (1998) alguns atributos devem ser seguidos:

a) utilização de Recursos, podendo ser classificados em comum, leve, média ou pesada;
b) tipos de Aplicações, onde uma carga pode ser agrupada de acordo com as aplicações a que pertence, como as WWW, FTP, Telnet, Mbone;
c) o objeto pode-se classificar em função do tipo tratado pelas aplicações, ou seja, HTML, imagens, sons, vídeos, texto, entre outros;
d) não funcional a carga pode ser agrupada de acordo com a função exercida;
e) no modo de processamento há uma classificação de três módulos básicos: O interativo que é um processo on-line; Transacional que também é um processamento on-line, podendo ser descrito como ordem de compra em um site. Já o lote é um processamento em *batch*, ou seja, apresenta-se como um agendamento no sistema Unix.

3.1.6 Construção do Modelo de Carga

A construção do modelo implica na determinação dos parâmetros de cada classe de componente, duas técnicas são as mais empregadas, a média e conjuntos.

3.1.6.1 Média

Um método simples para caracterizar os parâmetros de carga de trabalho é utilizar a técnica da média, que é um número simples que resume os valores dos parâmetros observados. Se $\{ x_1, x_2, x_3 ..., x_n \}$ são n valores observados de um dos parâmetros da carga de trabalho, a alternativa mais comum é a média aritmética, dada por:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Existem casos, onde a média aritmética não é a medida mais apropriada, neste caso, a mediana, moda, média geométrica ou média harmônica deverão ser usadas. A soma ou a média não tem significados se a freqüência das modas não é diferente de outros valores *n*, que podem ser especificados mais freqüentes (MENASCÉ; ALMEIDA, 1998).

3.1.6.2 Especificando as Medidas de Dispersão

O emprego de medidas como a média, para caracterizar uma carga de trabalho, deve ser acompanhado de medidas de dispersão, se existir uma grande variabilidade nos dados, essas, geralmente especificada pela variância, é denotada por S^2 e é calculada como segue:

$$S^2 = \frac{1}{n-1} \sum_{i+1}^{n} (x_i - \bar{x})^2$$

O desvio padrão S, que é a raiz quadrada da variância, encontra-se geralmente mais significativa porque aparece de forma mais expressiva na mesma unidade que a média, com isso razão entre o desvio padrão e a média é chamado de *Coeficiente de Variação* (C.O.V) (MENASCÉ; ALMEIDA, 1998).

Por exemplo, se ao longo de seis meses foram medidas em seis universidades a demanda por recursos devido à execução de vários programas, observando a média descrita na Tabela 3, onde os C.O.Vs dos valores medidos são bastante altos, indicando que a combinação de todos os programas dentro de uma classe não é uma boa idéia, portanto os programas deverão ser divididos dentro de várias classes. A Tabela 4 mostra a demanda média relativa ao uso de editores de texto, considerando os mesmos dados, só que agora os C.O.Vs são mais baixos.

Tabela 3. Caracterização da carga de trabalho usando valores médios

Dados	Média	Coeficiente de Variação
Tempo de CPU	2,19 segundos	40,23
Número de gravações diretas	8,20	53,59
Bytes escritos	10,21 kbytes	82,41
Número de leituras diretas	22,64	25,65
Bytes lidos	49,70 kbytes	21,01

Fonte: MENASCÉ, D; ALMEIDA, V. (1998)

Tabela 4. Características de uma sessão de edição média

Dados	Média	Coeficiente de Variação
Tempo de CPU	2,57 segundos	3,54
Número de escritas diretas	19,74	4,33
Bytes escritos	13,46 kbytes	3,87
Número de leituras diretas	37,77	3,73
Bytes lidos	36,93 kbytes	3,16

Fonte: MENASCÉ, D; ALMEIDA, V. (1998)

3.1.6.3 Conjuntos

Para os objetivos de análise da carga de um sistema, quando a quantidade de componentes é muito grande, é útil classificá-los em um pequeno número ou conjunto, de forma que os componentes dentro do conjunto sejam similares, e posteriormente pode-se tirar

um elemento de cada um para representar a classe e estudar os efeitos de possíveis alterações no sistema sobre a classe inteira.

3.1.6.4 Problemas com Conjuntos

Os conjuntos têm sido usados com sucesso para caracterização de cargas em vários ambientes, entretanto, pode-se conseguir conclusões conflitantes com os mesmos dados. Em geral é preciso minimizar a variância intragrupo e maximizar a intergrupos (MENASCÉ; ALMEIDA, 1998).

O emprego de conjuntos é melhor do que a escolha aleatória dos programas para a caracterização da carga de trabalho, contudo os resultados dos mesmos são altamente variáveis. Não existem regras padronizadas para a seleção dos parâmetros, medidas de distância, ou seja, uma escolha diferente de algum desses parâmetros levará a resultados totalmente diferentes e respectivamente também a conclusões distintas.

3.1.6.5 Cargas em Rajadas

A importância do tráfego em rajadas é uma grande influência sobre o desempenho. As requisições HTTP, podem apresentar, durante o período, taxas de pico que excedem a taxa média em 5 ou 10 vezes, o que pode facilmente ultrapassar a capacidade de atendimento do servidor da Web, e levar a degradação do *throughput* com uma pequena rajada (MENASCÉ; ALMEIDA, 1998).

Tornou-se evidente nessa seção as etapas que compõem a metodologia para a caracterização da carga de trabalho, tais como a definição da perspectiva de análise, identificação dos componentes básicos, definição do conjunto de parâmetros que melhor

identificam a carga, monitoração e coleta de dados, partição e a construção do modelo de carga.

3.2 ESTATÍSTICAS E DISTRIBUIÇÃO DE PROBABILIDADE

A caracterização da carga de trabalho contém valores quantitativos associados aos parâmetros do modelo que são baseados na análise dos dados de medição. Para alguns parâmetros dessa carga pode-se empregar as estatísticas simples como a média, a mediana e a variância. As distribuições das mesmas oferecem um modo mais comum de se capturar a variação de um parâmetro em um intervalo de valores.

3.2.1 Média, Mediana e Variância

Estatísticas como média, mediana e variância capturam as propriedades básicas de muitos parâmetros de carga de trabalho da Web, as características dos tamanhos de respostas podem ser representadas, como o tamanho médio, entretanto, a média não captura a variabilidade dos parâmetros da carga. Quando um parâmetro possui variabilidade alta, a média torna-se uma estatística não representativa de forma significativa, que pode ser distorcida por uma pequena quantidade de valores altos, com isso considere um servidor que gere cinco mensagens de resposta com tamanhos de 4.100, 4.700, 4.200, 20.000, e 4.000 bytes, diante disso, o tamanho médio da resposta é 7.400 bytes, mesmo assim, esse valor não oferece uma representação exata dos tamanhos de resposta típicos. Uma estatística alternativa é a *mediana,* metade dos recursos é menor, sendo que a outra parte é maior. Neste exemplo, o tamanho do recurso é de 4.200 bytes, que é uma indicação melhor do tamanho típico. Uma seqüência de 4.100, 4.700, 4.200, 4.800 e 4.000 bytes teria a mesma mediana de 4.200.

Outras estatísticas, como a variância ou o desvio-padrão, tentam quantificar a variação do parâmetro do valor médio, o que é indicado por uma pequena variância, tornando-se próximo do valor médio, enquanto uma grande variância pode ter valores que diferem da média, mas mesmo assim, essas estatísticas, não oferecem informações suficientes para a criação de uma carga de trabalho que represente o modo como os parâmetros variam na prática (KRISHNAMURTHY; REXFORD, 2001).

3.2.2 Distribuição de Probabilidade

As distribuições de probabilidade capturam o modo como o parâmetro varia por um grande intervalo de valores, várias distribuições de probabilidade foram estudadas e aplicadas à caracterização da carga de trabalho, sendo que cada uma pode ser representada por uma equação relativamente simples, com uma ou mais variáveis, sendo que a mais populares são as exponenciais, por meio da fórmula

$$F(x) = e^{-\lambda x}$$

com uma média de 1/λ, onde x pode assumir qualquer valor maior ou igual a 0 (KRISHNAMURTHY; REXFORD, 2001).

A caracterização de um sistema diversificado e em evolução como a Web, é muito difícil, podendo as medições coletadas em diferentes partes levar a outros resultados sobre as características da carga de trabalho, sendo assim a distribuição dos tamanhos da resposta entre clientes sem fio com pouca largura de banda, pode diferir de forma marcante dos mesmos com muita largura de banda. Em alguns casos, as mudanças em um parâmetro afetam algumas das variáveis de uma distribuição de probabilidade sem alterar a do sistema, embora o tempo de resposta médio possa variar entre os dois casos (KRISHNAMURTHY; REXFORD, 2001).

3.3 CARACTERÍSTICAS DA MENSAGEM HTTP

O protocolo HTTP faz a comunicação entre o cliente e o servidor por meio de mensagens, ou seja, o cliente envia e o servidor retorna ao mesmo com a solicitação, sendo que as propriedades básicas delas dão uma indicação de como o protocolo é utilizado na prática. As estatísticas sobre os métodos de pedido de códigos de resposta podem guiar a criação de cargas de trabalho reais da Web e identificar possíveis problemas com sites.

3.3.1 Métodos de Pedidos HTTP

O protocolo HTTP/1.1 possui um conjunto de métodos, que são utilizados na requisição de recursos, sendo estes os GET, HEAD, POST, OPTIONS, PUT, DELETE, TRACE, CONNECT.

Um pequeno número de métodos é responsável pela grande maioria dos pedidos HTTP, no entanto saber quais utilizar é útil para otimizar as implementações do servidor para os métodos mais comuns e desenvolver benchmarks reais para avaliar proxies e servidores da Web (KRISHNAMURTHY; REXFORD, 2001).

3.3.1.1 Características de Tráfego

O método GET é reconhecido por todos os servidores, sendo utilizado como método padrão para a requisição de recursos por meio do protocolo HTTP. Esse mesmo solicita ao servidor para que encontre e retorne como resposta ao cliente. Uma minoria de pedidos HTTP utiliza o método POST, normalmente para submeter dados em formulários, assim como as medições mostram um pequeno número de pedidos HEAD, que podem ser

gerados manualmente como parte de um servidor da Web operacional (KRISHNAMURTHY; REXFORD, 2001).

Os *sites* que possuem uma mistura incomum de métodos de pedido podem ter escolhas de desempenho diferentes, considerando que ele permite aos usuários enviarem e receberem mensagens de e-mail. Para submetê-la, o mesmo entra com texto em um formulário na janela *browser* e dá um clique em um botão para prosseguir com a mensagem, esse *site* pode receber um número relativamente grande de pedidos POST, e essas mensagens de pedido podem incluir uma grande quantidade de dados.

3.3.1.2 Tendências Futuras

Diversos *sites* da Web permitem que os usuários submetam formulários para várias finalidades, como a emissão de consultas para utilitários de pesquisa ou envio de e-mail, se os mesmos permitem aos usuários submeterem dados que se tornassem mais comuns, a proporção de pedidos POST aumentaria, além disso, um *browser* ou um *proxy* poderia usar o método HEAD para verificar se uma resposta em cache está ou não atualizada, antes de receber um pedido pelo recurso, o mesmo aconteceria com TRACE, com o surgimento de ferramentas para testar e depurar componentes da Web.

3.3.2 Código de Resposta HTTP

O HTTP define uma grande variedade de códigos de resposta. Uma parte importante da construção de um modelo realístico para as cargas de trabalho da Web é entender como os servidores normalmente respondem aos pedidos dos clientes, ou seja enviando em resposta a um pedido bem-sucedido, o código 200 é responsável por 75% a 90%

das respostas, outro mais comum é 304 *Not Modified*, normalmente responsável por 10% a 30% (KRISHNAMURTHY; REXFORD, 2001).

Diversos fatores são responsáveis pela grande variedade nos códigos de resposta de *sites*, sendo que o código de resposta 304 *Not Modified* aparece quando um cliente deseja validar uma cópia de um recurso em cache.

Os *sites* da Web com a maioria do conteúdo estático, como arquivos HTML e imagens embutidas, provavelmente retornarão um grande número de respostas 304 *Not Modified.*

Um grande número de resposta da classe de erro do cliente (4xx) pode indicar um problema com a organização de um *site* da Web ou outras páginas que possuem links de hipertexto apontando para elas, considerando o que acontece quando um administrador reorganiza os conteúdos em um *site*, alguns URLs podem não apontar mais recursos válidos no servidor, então esse ainda pode aparecer em outras páginas ou nos marcadores de páginas dos usuários, com isso um grande número de respostas 404 *Not Found* poderia sugerir, mostrando que muitos pedidos ainda usam os URLs antigos (KRISHNAMURTHY; REXFORD, 2001).

Certos códigos de resposta não surgem a menos que o cliente use cabeçalhos de pedido específicos. Considerando o código de resposta 206 *Partial Content* que é usado quando o servidor retorna um intervalo de bytes do recurso solicitado, esses podem se tornar mais comuns quando os *proxies* HTTP/1.1 forem mais utilizados, além disso, um *browser* pode invocar um manipulador que gere pedidos de intervalo. Por exemplo, um manipulador que apresenta arquivos PDF pode emitir pedidos de intervalo para acessar as páginas de um documento uma por vez, sendo que haja técnicas para melhorar o desempenho do *browser* e do *proxy* também poderiam aumentar a freqüência do código de resposta 206 *Partial Content* (KRISHNAMURTHY; REXFORD, 2001).

A freqüência das respostas de redirecionamento 302 *Found* varia de um *site* para outro, alguns utilizam as respostas de redirecionamento como um meio de equilibrar a carga de pedidos por uma coleção de réplicas, sendo assim a mesma instrui o cliente a repetir o pedido para esse servidor, sendo que o tamanho e a escala cada vez maiores da Web sugerem que os servidores replicados se tornarão cada vez mais comuns, especialmente para *site*, apresentando-se mais populares (KRISHNAMURTHY; REXFORD, 2001).

3.4 CARACTERÍSTICAS DO RECURSO DA WEB

Os recursos variam em termos de tamanho, popularidade e freqüência com que são alterados, além disso, páginas HTML mudam em termos dos números de referência embutidas que possuem, uma parte importante da modelagem das cargas da Web é entender as características, tais como o tipo de conteúdo, tamanho do recurso, da resposta, popularidade do recurso, freqüência de modificação, localidade temporal e o número de recursos embutidos (KRISHNAMURTHY; REXFORD, 2001).

3.4.1 Tipos de Conteúdo

As estatísticas sobre os tipos de conteúdo oferecem uma indicação dos tipos de dados disponíveis na Web, esses também possuem um relacionamento direto com outros parâmetros básicos da carga de trabalho, como o tamanho do recurso e a freqüência de modificação, além disso, certos tipos de conteúdo, como texto, são mais fáceis de compactar para reduzir o tamanho das mensagens de resposta, enquanto outros, como imagens, normalmente são codificados em um formato compactado (KRISHNAMURTHY; REXFORD, 2001).

Esses tipos de conteúdos incluem documentos como *PostScript* e PDF, softwares como *JavaScript* ou *applets* Java, e dados de áudio e vídeo, no entanto, pode variar muito de um site para outro (KRISHNAMURTHY; REXFORD, 2001).

3.4.2 Tamanhos de Recurso

A distribuição exata dos tamanhos de recurso varia entre servidores e com o tempo, a análise das medições da Web mostram várias características comuns, essas mesmas afetam os requisitos de armazenamento no servidor de origem e o *overhead* do *caching* de recursos nos *browsers* e *proxies*, além disso, a carga sobre a rede e a latência na entrega da mensagem de resposta sofreram uma influência (KRISHNAMURTHY; REXFORD, 2001).

Em geral o tamanho médio do recurso é relativamente pequeno, embora essa proporção seja muito grande. Texto, HTML e imagens costumam ser menores do que outros tipos de conteúdo, como áudio e vídeo, sendo que arquivos HTML possuem uma mediana de 2 Kb, muito menor do que o tamanho médio de 4 a 8 Kb, isso sugere uma variação entre os tamanhos.

Embora a mediana calcule o tamanho de um recurso típico, a distribuição de probabilidade oferece uma indicação melhor da variabilidade nos tamanhos de recursos, que é útil para a decisão de como alocar memória ou espaço de disco em um *proxy* ou servidor. A alta variabilidade no tamanho do recurso normalmente é capturada pela distribuição de Pareto, onde:

$$F(x) = \left(\frac{\kappa}{x}\right)^{a}, x \geq \kappa$$

Com um parâmetro de forma α e outro de escala κ, possui uma média de $\kappa\alpha/(\alpha - 1)$ para $\alpha > 1$. A exponencial também é bastante usada e muito menos variável do que a de Pareto, sendo

que, essa, somente 60% dos recursos são menores do que 1, embora os outros 40% sejam maiores. A média da distribuição é muito grande para valores de α próximos a 1, tornando-a uma medida difícil de se usar na caracterização de tamanhos de recurso da Web, apesar de ser precisa para recursos maiores. O corpo é melhor representado por um lognormal, nesta o logaritmo de x possui uma distribuição normal, ela também captura com precisão as características dos recursos da Web e junta a lognormal e de Pareto, que modelam todo o intervalo dos tamanhos de recursos (KRISHNAMURTHY; REXFORD, 2001).

3.4.3 Tamanhos de Resposta

Na análise de desempenho do servidor e da rede, o tamanho das mensagens de resposta é um fator mais importante, assim como o número de bytes e de largura de banda consumida para satisfazer um pedido do cliente.

Os tamanhos de resposta diferem dos tamanhos de recurso por diversos motivos, primeiramente, algumas mensagens de resposta HTTP não transferem um recurso, com isso as respostas 204 *No Content* e 304 *Not Modified* não possuem um corpo de mensagem, em segundo, alguns recursos da Web nunca são solicitados e, portanto, não contribuem para o conjunto de mensagens de resposta, ou seja (KRISHNAMURTHY; REXFORD, 2001).

3.4.4 Popularidade do Recurso

A popularidade é medida em termos da proporção dos pedidos que acessam um determinado recurso, e os recursos são classificados da maior para a menor popularidade, então a *função de massa de popularidade*, P(r), captura a proporção de pedidos direcionados

para cada um desses recursos, considerando um *site* da Web com 100 recursos. Se todos os 100 recursos fossem igualmente populares, então P(r) = 0,01 para r = 1, 2, ..., 100.

Curvas mais íngremes como o gráfico da Figura 15, significam que alguns recursos são mais populares. No exemplo, o recurso mais popular é responsável por quase 10% dos pedidos (KRISHNAMURTHY; REXFORD, 2001).

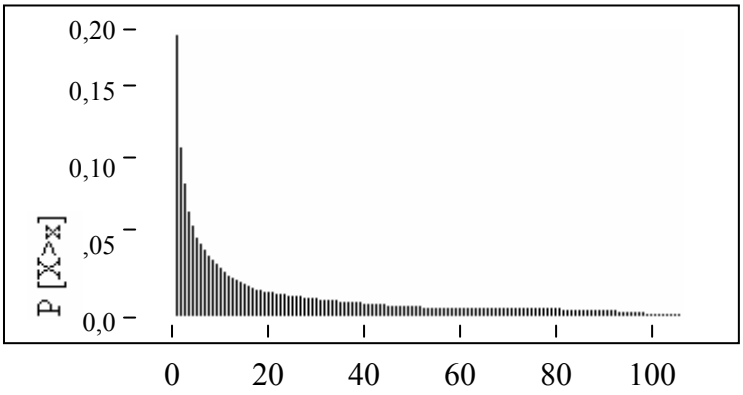

Figura 15. Lei de Zipt
Fonte: KRISHNAMURTHY, B; REXFORD, J. (2001)

A proporção de pedidos por um recurso é inversamente proporcional à sua classificação. A distribuição segue a *Lei de Zipf*, onde:

$$P(r) = \kappa r^{-1}$$

onde κ é uma constante de proporcionalidade que garante que P(r) chegue ao total 1. A lei de Zipt se aplica a uma grande variedade de situações, como a popularidade de diferentes palavras em um documento. Geralmente, uma distribuição de Zipt possui a forma:

$$P(r) = \kappa r^{-c}$$

Valores menores de *c* correspondem a diferenças menores em popularidade entre conjunto de recursos. O caso extremo de *c* = 0 corresponde a todos os recursos tendo a mesma popularidade (KRISHNAMURTHY; REXFORD, 2001).

Resultados semelhantes são obtidos para *sites* da Web, exceto para um pequeno número que recebe a maioria dos pedidos. A inclinação na popularidade possui implicações importantes para o seu desempenho.

3.4.5 Mudanças no Recurso

A freqüência das mudanças e as diferenças entre as sucessivas versões de uma resposta possuem implicações importantes em uma série de aplicações da Web. Onde, muitos utilitários de pesquisa dependem de *spiders* para apanhar cópias de páginas para indexação. Saber com que repetição os recursos mudam em um *site* e o significado dessas mudanças, podem ajudar a determinar quando as páginas da Web devem ser apanhadas pelo *spider*, da mesma forma as modificações nos recursos afetam o desempenho do *caching*. Observa-se que os recursos que mudam com menos freqüência tem preferência no *caching* ou podem ser revalidados com o servidor de origem. No entanto, mudanças dos recursos variam com base no tipo de conteúdo, sendo que na maioria dos *sites*, as imagens não mudam com muita freqüência (KRISHNAMURTHY; REXFORD, 2001).

No futuro, novos tipos de conteúdo poderão surgir, sendo mudados com mais freqüência do que os tipos de conteúdos existentes, porém essas tendências afetariam as aplicações, como *spiders* e *caching*, que são influenciadas pelas mudanças nos recursos da Web.

3.4.6 Localidade Temporal

A localidade temporal captura a probabilidade de que um recurso solicitado seja pedido novamente no futuro próximo. Quando uma seqüência de pedidos apresenta alta

localidade temporal, existe uma chance maior de que recursos solicitados já residam na memória principal do servidor de origem ou na cache do *proxy*. A captura exata da localidade temporal em um fluxo de referências é uma parte importante da modelagem da carga de trabalho de um *proxy* ou servidor.

A localidade temporal de um recurso pode ser caracterizada pela consideração da distribuição de distâncias de pilhas por uma seqüência inteira de pedidos, sendo que a maioria desses acessam um recurso que foi solicitado (KRISHNAMURTHY; REXFORD, 2001).

3.4.7 Número de Recursos Embutidos

Esses recursos embutidos incluem imagens, programas *JavaScript* e outros arquivos HTML que aparecem como frames na página da Web principal. O número de referências embutidas em uma página da Web possui um impacto significativo sobre a carga do servidor e da rede, sendo assim o cliente emite um pedido HTTP separado para cada recurso embutido, a menos que uma cópia em cache esteja disponível.

O modelo de carga de trabalho que não inclua o *download* automático de recursos embutidos não capturaria os aumentos repentinos na carga sobre o servidor e a rede, além disso, afeta a eficiência das conexões, fazendo com que um grande número desses recursos aumente a probabilidade de que uma única conexão TCP trate de vários pedidos HTTP (KRISHNAMURTHY; REXFORD, 2001).

3.5 CARACTERÍSTICAS DO COMPORTAMENTO DO USUÁRIO

A chegada de usuários aos *sites* da Web, o número de páginas que baixam e o tempo que esperam entre os *downloads* sucessivos contribuem para os padrões de tráfego

visto nos *proxies* e servidores, essas características da carga de trabalho da Web também dependem do comportamento dos usuários.

3.5.1 Chegadas de Sessão e Pedido

A carga de trabalho introduzida por um único usuário pode ser modelada em três níveis: sessão, clique e pedido. A sessão do usuário começa com o primeiro pedido e termina após o último pedido de inatividade. Cada clique corresponde a uma ação do usuário, como a seleção de um link de hipertexto, a submissão de um formulário ou a digitação de um URL e dispara o browser para emitir um pedido HTTP por um recurso, seguido por pedidos gerados automaticamente para os recursos embutidos e referenciados nessa página (KRISHNAMURTHY; REXFORD, 2001).

Do ponto de vista do servidor, cada chegada de sessão traz um novo usuário ao *site*, fazendo com que cada clique dispare uma rajada de pedidos HTTP para o servidor tratar, podendo estabelecer uma nova conexão TCP existente.

3.5.2 Cliques por Sessão

Se um usuário fizer o download de um grande número de páginas, a carga introduzida por cada sessão envolveria diversas transferências por um período de tempo, por outro lado, se a maioria dos usuários acessa apenas uma página, cada sessão envolveria a transferência de uma única página da Web e suas imagens embutidas. Se cada sessão do usuário consistir em um único clique, o servidor poderia ter que transferir o conjunto inteiro de imagens embutidas para cada página.

Uma pequena proporção dos visitantes de um site é responsável pelo núcleo dos pedidos, quando esses usuários visitam um site, o grande número de pedidos pode degradar o desempenho visto por outros usuários logo, um modelo de carga de trabalho real precisa capturar o impacto desses usuários pesados sobre a rede e o servidor (KRISHNAMURTHY; REXFORD, 2001).

3.5.3 Tempo entre Chegadas de Pedido

Um browser normalmente emite pedidos de recursos embutidos enquanto recebe e analisa o arquivo HTML principal, no entanto, o tempo entre os cliques sucessivos depende do comportamento do usuário, sendo que o intervalo entre o downloading de uma página e suas imagens embutidas e próximo clique do usuário é denominado *tempo de pensamento ou tempo de silêncio* (KRISHNAMURTHY; REXFORD, 2001).

Em geral, o tráfego da Web apresenta variabilidade em várias dimensões, sendo assim o tempo típico entre cliques varia de um site para outro, embora a maioria dos tempos entre pedidos seja de menos de 60 segundos (s). Os estudos de medição mostraram que os tempos de pensamento seguem uma distribuição de Pareto, com α em torno de 1,5s. Esses não são tão variáveis quanto os tempos de resposta, que têm α na faixa de 1,0 a 1,5s (KRISHNAMURTHY; REXFORD, 2001).

O tráfego da Web consiste na superposição de várias seqüências de períodos ligado/desligado, cada um correspondente a um usuário diferente, como resultado, a carga nos servidores e na rede exibe um fenômeno conhecido como *auto-semelhança*, em que o tráfego varia bastante em uma série de escalas de tempo, desde microssegundos até vários minutos, possuindo implicações importantes para o modo como os servidores e componentes da rede são avaliados (KRISHNAMURTHY; REXFORD, 2001).

3.6 APLICANDO MODELOS DE CARGA DE TRABALHO

A geração do tráfego da Web a partir do modelo envolve a criação de um fluxo de pedidos HTTP de acordo como os diversos parâmetros de carga de trabalho e a aplicação desses pedidos a *proxies* e servidores operacionais, através de um estudo mais detalhado das características de carga de trabalho da Web podendo levar à criação de um modelo para a avaliação dos protocolos e componentes de software da Web.

3.6.1 Combinando Parâmetros de Carga de Trabalho

A distribuição da probabilidade para o tamanho dos recursos HTML pode ser usada para determinar o número de bytes em cada um dos arquivos, sendo que a distribuição mediante o número de recursos embutidos pode ser usada para associar a página. No fim, cada página da Web no servidor consiste em um arquivo HTML com um determinado tamanho e popularidade, e que está associado a algum número de recursos embutidos.

Tabela 5. Distribuições de probabilidade nos modelos de carga de trabalho da Web

Distribuição	Parâmetros de carga de trabalho
Exponencial	Tempos entre chegadas de sessão
Pareto	Tempo de resposta (cauda da distribuição) Tamanhos de recurso (cauda da distribuição) Número de imagens embutidas Tempos entre chegada de pedido
Lognormal	Tamanhos de resposta (corpo da distribuição) Tamanhos de recurso (corpo da distribuição) Localidade temporal
Zipf	Popularidade do recurso

Fonte: KRISHNAMURTHY, B; REXFORD, J. (2001).

A geração de uma carga de trabalho sintética requer a integração das características do recurso e a temporização dos pedidos do cliente, podendo este ser associado a um URL específico com base na distribuição de popularidade, no entanto, essa técnica não captura a localidade temporal dos pedidos sucessivos para o mesmo recurso. Outro aspecto importante é modelar os recursos que são modificados no servidor para capturar o impacto desses nos tamanhos, juntamente com uma mistura realista de tráfego de validação de cache, onde, os tempos de modificação de recurso poderiam vir de uma distribuição de probabilidade, e o recurso passar a ser alterado no servidor (KRISHNAMURTHY; REXFORD, 2001).

3.6.1.1 Validando o Modelo de Carga de Trabalho

A necessidade de validação de modelos surge em uma grande variedade de aplicações, onde sempre que o desempenho de um sistema é avaliado com entrada sintética ou com modelo abstrato, a validade dos resultados entra em questão. A validação do modelo pode ser difícil, demorada ou simplesmente desconsiderada, em geral, um modelo de carga de trabalho é construído para uma finalidade específica, como avaliar o *throughput* do servidor e a latência percebida pelo usuário (KRISHNAMURTHY; REXFORD, 2001).

Os modelos de carga de trabalho sintéticos também são usados para o teste de servidores por diversos cenários que poderiam não ter acontecido na prática, neste caso, o teste contra uma carga de trabalho real pode ser possível.

3.6.1.2 Gerando o Tráfego Sintético

O teste de um servidor de alto nível requer a geração de uma alta de pedidos de um grande número de cliente sintéticos, com cada cliente emitindo pedidos com base no modelo de carga de trabalho.

A geração de tráfego sintético oferece uma oportunidade para se avaliar um proxy ou servidor de uma forma controlada, onde a coleta de estatísticas de desempenho é uma parte terminante desse processo, no entanto, a medição e o registro dessas medidas de desempenho pode interferir com a operação do cliente e do servidor, pois o cliente sintético pode registrar o atraso no recebimento da resposta do servidor a um pedido HTTP, e o servidor pode registrar as medidas de desempenho, como a carga de processamento ou o número de conexões TCP com o tempo (KRISHNAMURTHY; REXFORD, 2001).

O desempenho da Web depende da interação entre o comportamento do usuário, as características do recurso, a carga do servidor e a dinâmica da rede, porém a captura de todos esses fatores em um modelo de carga de trabalho é extremamente difícil na prática, no entanto, as cargas de trabalho sintéticas ajudam a focalizar a necessidade de se avaliar e comparar os componentes de software da Web de uma forma controlada (KRISHNAMURTHY; REXFORD, 2001).

4 ANÁLISE DE LOGS E SOFTWARES UTILIZADOS E A CARACTERIZAÇÃO DA CARGA DE TRABALHO DE UM SERVIDOR WEB

Neste capítulo será descrito como foi aplicada a caracterização da carga de trabalho em arquivos de logs, os quais se encontram disponíveis livremente na Internet, em seu formato bruto, com a finalidade de demonstrar como podemos extrair as principais informações da carga de um servidor Web em um ambiente simulado. Com isso, utilizamos alguns softwares para esses propósitos, assim como a descrição das características de cada um deles e também suas funcionalidades.

Etapas realizadas para o desenvolvimento deste capítulo:

a) escolha do software do servidor Web;

b) levantamento e busca de softwares para analise de logs do servidor Apache;

c) busca de software para gerar cargas sintéticas em servidores Web;

c) busca de logs do servidor apache disponíveis na Internet;

d) etapas da caracterização da carga de trabalho, análises e resultados.

4.1 ESCOLHA DO SOFTWARE SERVIDOR WEB

A escolha do software do servidor Web, utilizou-se o Apache por ser confiável, estável, seguro, sua distribuição é livre, código fonte aberto, possui versões que rodam em vários sistemas operacionais e também por ser o mais utilizado no mundo. Segundo o Site da NetCraft, um levantamento feito em outubro de 2009, o Apache é utilizado em 46,90% de todos o servidores disponíveis no mercado. A Figura 16 mostra o nível de utilização do Apache entre outros servidores Web.

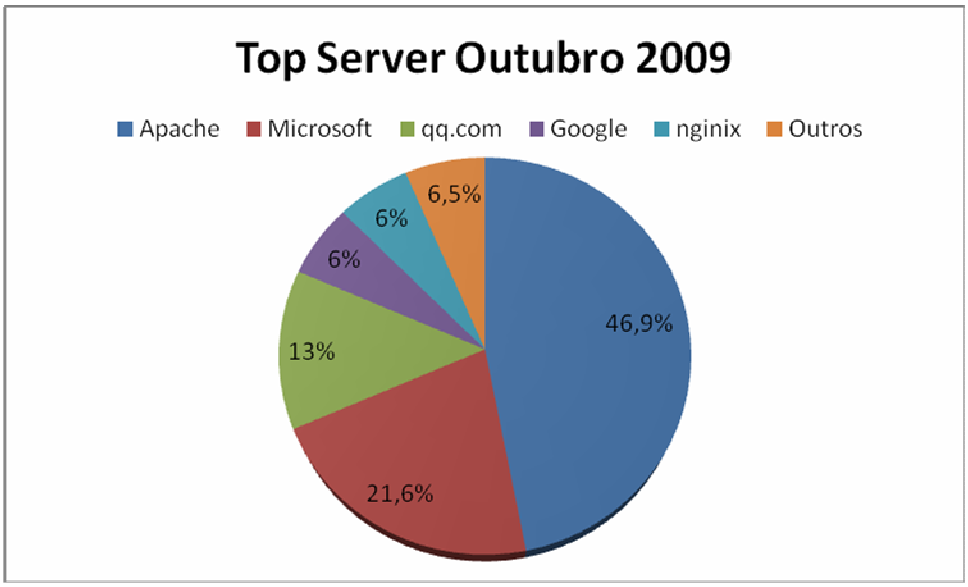

Figura 16. Utilização do servidor Apache
Fonte: Netcraft, 2009.

4.1.1 Servidor Apache

O servidor Apache foi projetado pelo *Apache Group,* com base no código do projeto inicial da *National Center for Super Computing Applications* (NCSA), que teve início no ano 1995. Após vários testes e correções, fizeram sua primeira distribuição gratuita na versão 0.6.2 e logo depois a versão 1.0 nesse mesmo ano.

O *Apache Group* ganhou força nos anos seguintes com ajuda de vários webmasters que formaram um grupo de desenvolvedores do projeto, operando por meio da Internet e sem fins lucrativos.

O servidor Web Apache é implementado com um projeto modular, com código fonte aberto e distribuição gratuita. É desenvolvido por programadores com alguma experiência na linguagem C ou Perl, que podem contribuir com módulos que executam alguma função específica.

4.1.2 Características e Recursos do Apache

Um dos recursos oferecido pelo Apache e que o torna o servidor mais utilizado é a vantagem de ser um sistema multiplataforma. Existem versões que rodam nos sistemas operacionais Unix (Linux, FreeBSD, Solaris, entre outros), sistemas operacionais Windows (Windows 9X, 2000, NT, XP, Vista) e entre alguns outros sistemas operacionais.

Entre as várias características do servidor Web Apache, pode-se citar as mais importantes: suporte a protocolo HTTP 1.1; configuração primária em arquivos modo texto; suporte para autenticação de HTTP; suporte para host virtuais; suporte para CGI 1.1 e FastCGI; linguagem Perl integrada; suporte a criação de scripts PHP; suporte para servlets Java; status do servidor e logs; suporte para SSI e SSL e por fim, servidor de *proxy* integrado.

4.2 BUSCAS DE FERRAMENTAS PARA ANÁLISE DE LOGS DO SERVIDOR WEB APACHE

Durante a realização desse trabalho, utilizou-se ferramentas para análises dos logs do Apache. Essas foram utilizadas para a extração das principais informações que compõe uma requisição HTTP. Já que por meio do arquivo de log do servidor é possível extrair as principais informações para a caracterização da carga de trabalho. Há um vasto número de ferramentas desse gênero e dessas foram testadas várias, mas a maioria delas não coletavam todas as informações necessárias para análise das cargas. Então à opção por utilizar a que atendesse todos os requisitos e que melhor representasse graficamente os resultados.

4.2.1 Ferramentas Utilizadas para Análise dos Logs

As ferramentas que apresentaram um melhor resultado tanto na parte de extração quanto na representação gráfica dos resultados, foram as ferramentas AWStats, AnalisadorLogsApache e a Webalizer.

4.2.1.1 Ferramenta AWStats

AWStats é uma ferramenta gratuita e de grande utilidade, desenvolvida na linguagem Perl, essa é utilizada para análise das informações dos log. Gera estatísticas de serviços Web, streaming, FTP e-mail e disponibiliza as informações graficamente em formato HTML.

A ferramenta é de fácil utilização apenas configurando o arquivo awstats.conf, indicando o caminho do log do servidor, e configurando o formato que irá utilizar. Essa ferramenta roda em mais de uma plataforma e aceita formatos de logs de diferentes servidores Web, desde que o esteja de acordo com o W3C. Essa ferramenta exige que a linguagem perl esteja instalada na máquina.

A Figura 17 ilustra o gráfico gerado pela ferramenta AWStats com informações capturadas de um arquivo de log de um servidor real. As informações dos gráficos estão detalhadas na Tabela 6, que são do histórico mensal de abril e maio de 2009.

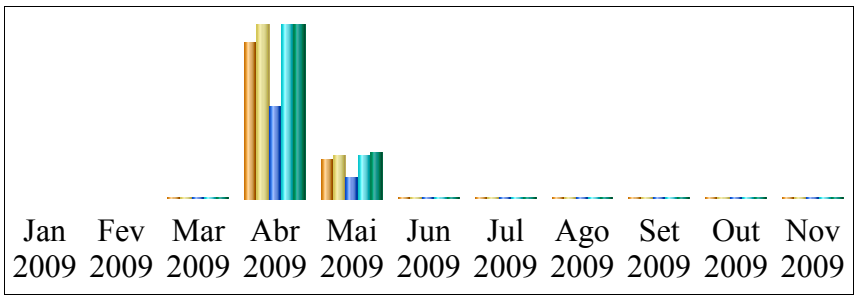

Figura 17. Histórico mensal gerado pela ferramenta AWStats

Tabela 6. Gráfico do histórico mensal gerado pela ferramenta AWStats

Mês	Visitantes únicos	Número de visitas	Páginas	Hits	Bytes
Abr 2009	7940	9248	55647	111220	12.69 GB
Mai 2009	2008	2232	13031	26705	3.33 GB
Total	9948	11480	68678	137925	16.02 GB

A ferramenta também consegue capturar informações como o *browser* e sistema operacional do cliente, para tal informações necessita que o log analisado esteja em seu formato completo (*Custom Log Format*). A Tabela 7 indica os navegadores utilizados pelos clientes por ordem de maior porcentagem.

Tabela 7. Navegadores utilizados pelos clientes

	Browsers	Grabber	Hits	Por cento
	MS Internet Explorer	Não	70304	63,2%
	Firefox	Não	31202	28,0%
	Google Chrome	Não	4328	3,8%
	Lynx	Não	4069	3,6%
	Opera	Não	535	0,4%
	Safari	Não	425	0,3%
	Mozilla	Não	155	0,1%
	Desconhecido	?	120	0,1%
	Netscape	Não	33	0%
	Samsung (PDA/Phone browser)	Não	27	0%
	Outros visitantes	?	22	0%

Uma das funcionalidades mais importantes analisadas no software AWStats é que se consegue extrair dos logs os tipos de arquivos solicitados em uma requisição HTTP. Função que a grande maioria dos softwares testados não apresentava.

Tabela 8. Tipos de arquivos

Tipos de arquivo		Hits	Por cento	Bytes	Por cento
gif	Image	41954	37.7 %	51.27 MB	0,3%
php	Dynamic PHP Script file	40787	36.6 %	8.42 GB	66,3%
css	Cascading Style Sheet file	10344	9.3 %	10.01 MB	0%
html	HTML or XML static page	8687	7.8 %	3.89 GB	30,6%
cgi	Dynamic Html page or Script file	4546	4%	2.37 MB	0%
jpg	Image	2129	1.9 %	25.75 MB	0,1%
htm	HTML or XML static page	1520	1.3 %	300.12 MB	2,3 %
png	Image	745	0.6 %	1.41 MB	0%
swf	Macromedia Flash Animation	394	0.3 %	2.05 MB	0%
php3	Dynamic PHP Script file	103	0%	246.01 KB	0%
js	JavaScript file	7	0%	35.34 KB	0%
exe	Binary runtime	2	0%	2.42 MB	0%
jar	Archive	1	0%	8.85 KB	0%
doc	Document	1	0%	81.00 KB	0%

4.2.1.2 Ferramenta AnalisadorLogsApache

Essa ferramenta foi desenvolvida por Silva (2006) e foi utilizado para a análise dos pedidos HTTP armazenado no log do servidor Apache. Essa foi de extrema importância para cruzamentos de dados, possibilitando uma melhor visualização dos objetos que componha os pedidos de requisição e conseqüentemente uma maior precisão nos resultados obtidos. De várias ferramentas testadas, essa foi a única que possibilitou, o cruzamento de dados e a única que capturou os métodos de acesso e a versão do protocolo HTTP.

A ferramenta é de simples funcionalidades mas de grande utilidade. Seu desenvolvimento foi na linguagem C. Na Figura 18 é demonstrado as informações analisadas de um arquivo de log de um servidor real.

Universidade de São Paulo - ICMC-USP...
Desenvolvido por Luis Henrique Castilho da Silva...

Quantidade de requisições por método de acesso

Metodo	Qtde requisições	%	Bytes Tranf	%
GET	819242	99,89	69657038500	100,00
POST	614	0,07	138938	0,00
OPTIONS	237	0,03	58710	0,00
HEAD	77	0,01	17341	0,00
CONNECT	10	0,00	2310	0,00
PUT	3	0,00	752	0,00
Total:	820183		69657256551	

Quatidade de requisicoes e bytes discriminadas por TIPO DE REQUISICAO... Metodo [GET]

Tipo Req.	Qtde requisicoes	%req_metodo	%req_total	Bytes	%metodo	%total
200	657775	80,29	80,20	52900513717	75,94	75,94
206	114862	14,02	14,00	16745818828	24,04	24,04
404	34685	4,23	4,23	7863127	0,01	0,01
301	6346	0,77	0,78	1580836	0,00	0,00
403	4941	0,60	0,60	1121961	0,00	0,00
400	633	0,08	0,08	140031	0,00	0,00
Total:	819242			69657038500		

Figura 18. Informações analisadas com a ferramenta AnalisadorLogsApache

4.2.1.3 Webalizer

Ferramenta desenvolvida 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. As estatísticas e seus resultados são representados em forma de gráficos em documentos HTML.

Sua instalação e configuração são bastante simples e seu código fonte está disponível gratuitamente. Para rodar essa ferramenta basta salvar os arquivos baixados em uma pasta, e após isso, configurar o arquivo webalizer.conf que está disponível em modo

texto, apontado o caminho do servidor Apache, do arquivo log a ser analisados e a pasta onde serão geradas as estatísticas que poderão ser acessadas pelo navegador Web.

Além de informações como histórico de visitantes, números de páginas e outras informações comuns, a ferramenta consegue capturar os códigos de resposta do servidor, funcionalidade que em muitas ferramentas pesquisadas não possuíam. A Tabela 9 mostra os códigos de resposta de uma requisição HTTP, analisada de um arquivo de log de um servidor Web real.

Tabela 9. Informações do código resposta do servidor HTTP

Hits By Response Code		
Code 200 - Ok	78,29%	47881
Code 206 - Partial Content	15,51%	9487
Code 301 - Moved Permanently	0,78%	478
Code 400 - Bad Resquest	0,12%	73
Code 403 - Forbidden	0,95%	581
Code 404 - Not Found	4,33%	2651
Code 405 - Method Not Allowed	0,00%	1
Code 501 - Not Implemented	0,01%	4

4.3 FERRAMENTAS PARA GERAR CARGAS SINTÉTICAS EM SERVIDORES WEB

Outras ferramentas utilizadas foram para gerar carga sintética para o servidor Web Apache. Dessa ferramenta é possível analisar, por exemplo, o uso da CPU, memória, dados que não pode ser analisados a partir somente dos logs.

4.3.1 Webserver Stress Tool

Ferramenta de stress, desenvolvida pela empresa Paessler, essa ferramenta gera carga sintéticas para servidores Web. Essa ferramenta foi escolhida por ser de fácil configuração e por gerar vários gráficos, como por exemplo, uso da CPU, da memória,

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, planejamento de capacidade entre outros.

Essa ferramenta gera cargas de trabalho baseadas em cargas reais, por esse motivo as informações ficam armazenada no log do servidor Web. Além dos gráficos já gerados pela ferramenta, pode-se utilizar o log para uma análise mais detalhada. Mas como esse estudo utilizou logs de servidores HTTP reais, essa ferramenta foi utilizado somente para verificar graficamente o uso dos recursos de máquina. Já que somente com as informações contidas nos logs não é possível verificar esses tipos de dados.

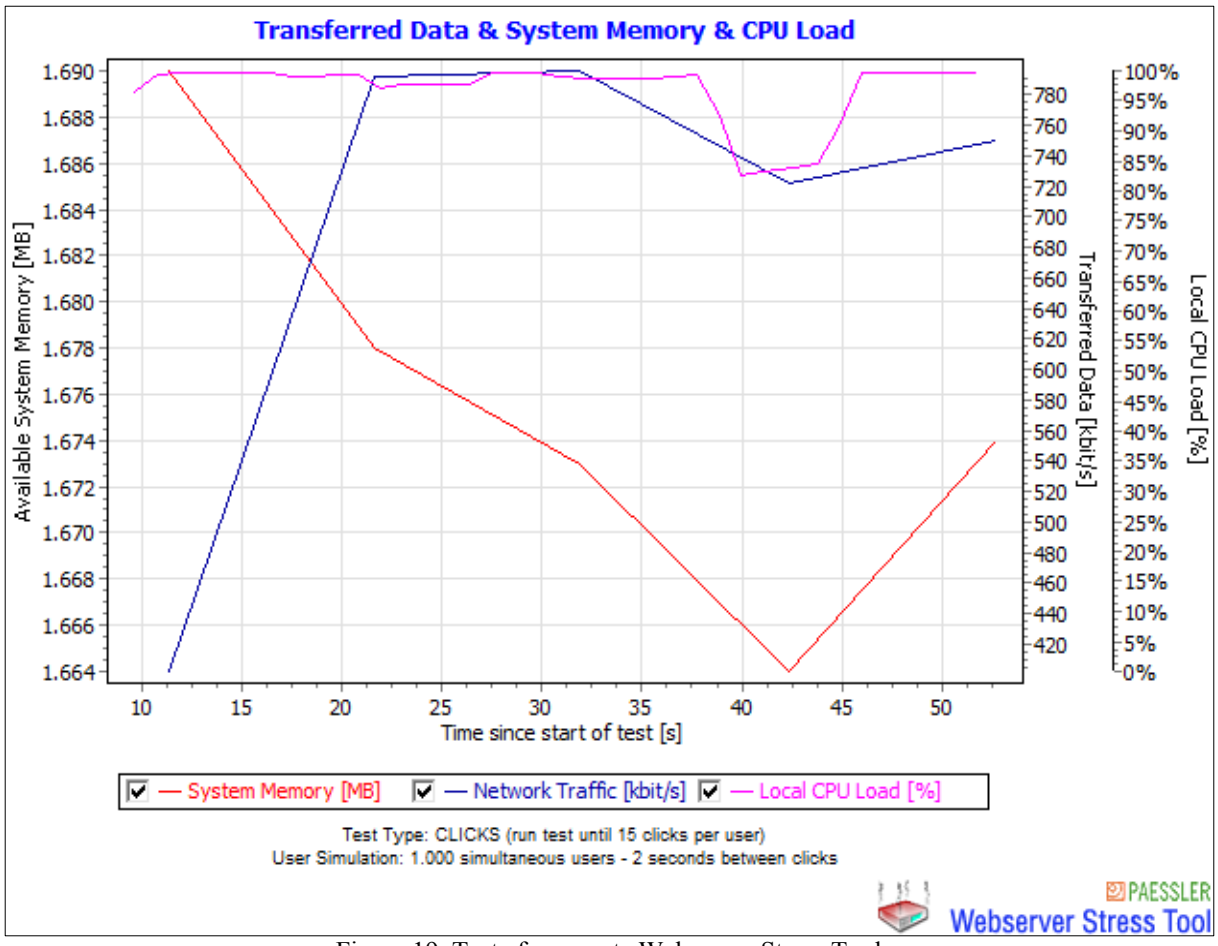

Figura 19. Teste ferramenta Webserver Stress Tool

No gráfico acima é ilustrado um teste rodado na ferramenta Webserver Stress, a linha rosa indica o uso da CPU, note que o uso chegou bem próximo de 100%, a azul indica o tráfico e a laranja o uso da memória. Com essa ferramenta é possível definir o número de usuários, quantidade de requisições por usuário e tempo entre as requisições. Com isso, é possível criar uma sobrecarga no servidor, indicado para tratar teste de desempenho e planejamento de capacidade. No gráfico da Figura 19 foi gerada uma simulação com 1.000 usuários, com 15 cliques por usuário, (cada clique representa uma requisição), o tempo de delay entre um clique e outro, foi definido em dois segundos, mas pode ser definido como zero, isso implicaria numa sobrecarga no servidor.

4.3.2 Apache Bench

Apache Bench ou Apache Benchmark é uma ferramenta distribuída junto com o servidor Web Apache. Usada para gerar cargas de trabalho sintéticas para teste de desempenho. Para utilizar o Apache Bench, basta apenas utilizar o comando ab. Abaixo segue exemplo de como rodar um teste.

```
ab -n 2500 -c 500 http://127.0.0.1:80/
```

No exemplo -n 2500 indica o número de requisição (2.500 request), -c 500, indica a concorrência, ou seja, o número de usuários simultâneos, no caso 500, em seguida vem o endereço inicial e a porta usada pelo servidor.

Abaixo segue o resultado do teste simulado descrito acima, o apache usa threads para simular diferentes usuários acessando o site simultaneamente.

```
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 127.0.0.1 (be patient)
Completed 250 requests
Completed 500 requests
Completed 750 requests
Completed 1000 requests
Completed 1250 requests
Completed 1500 requests
Completed 1750 requests
Completed 2000 requests
Completed 2250 requests
Completed 2500 requests
Finished 2500 requests

Server Software:        Apache/2.2.14
Server Hostname:        127.0.0.1
Server Port:            80

Document Path:          /
Document Length:        44 bytes

Concurrency Level:      500
Time taken for tests:   93.844 seconds
Complete requests:      2500
Failed requests:        2
   (Connect: 2, Receive: 0, Length: 0, Exceptions: 0)
Write errors:           0
Total transferred:      822500 bytes
HTML transferred:       110000 bytes
Requests per second:    26.64 [#/sec] (mean)
Time per request:       18768.750 [ms] (mean)
Time per request:       37.538 [ms] (mean, across all concurrent requests)
Transfer rate:          8.56 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   32  119.6      0    1063
Processing:  2000 17755 4491.1  19828   21469
Waiting:     2000 12681 3977.5  12203   20906
Total:       2000 17787 4492.2  19844   21469

Percentage of the requests served within a certain time (ms)
  50%  19844
  66%  19922
  75%  20016
  80%  20250
  90%  20375
  95%  20422
  98%  20906
  99%  20938
 100% 21469 (longest request)
```

4.4 OUTRAS FERRAMENTAS RELACIONADAS

Além da ferramenta Webserver Stress também foram testadas outras ferramentas e estudas o funcionamento das mesmas.

4.4.1 W4gen

Ferramenta desenvolvida por Silva (2006) teve como estudo para obtenção do titulo de Mestre em Ciências de Computação e Matemática Computacional pela USP de São Carlos. Ferramenta que gera cargas sintéticas para modelos de servidores, podendo ser modificada para cada caso de estudo. Possui várias funções de distribuição de probabilidade possibilitando produzir cargas bem próximas das cargas reais. Permite ao usuário selecionar modelos pré-definidos ou criar um modelo próprio.

Ferramenta desenvolvida em linguagem Java e utiliza a biblioteca PSOL (*Probability/Statistic Object Library),* é uma biblioteca para representar os conceitos e técnicas da probabilidade e estatística. Ferramenta poderosa para gerar cargas sintéticas possibilitando o usuário a modificar parâmetros da carga, obtendo uma reflexão direta no desempenho do servidor (SILVA, 2006).

4.4.2 SPECweb

É um produto de software para *benchmark* desenvolvido pela SPEC, projeto para medir a capacidade de um sistema de atuar como servidor Web. SPECweb99 mede o número máximo de conexão simultânea que um servidor Web é capaz de aceitar enquanto atende a requisitos específicos de taxas de processamento e de erros. A carga de trabalho padrão inclui HTML gerada estática e dinamicamente, esse benchmark pode rodar em diferentes

plataformas de sistema (MENASCÉ; ALMEIDA, 2002). Essa ferramenta foi somente pesquisada para relacionar entre as ferramentas disponíveis para teste em servidores.

4.4.3 JMETER

Ferramenta também utilizada para gerar carga em sistemas, desenvolvida em Java pelo projeto Jakarta da *Apache Software Foundation*.

Essa ferramenta foi testada, porém seu funcionamento e configuração são de alta complexidade, por esse motivo não foi optada em utilizar essa ferramenta. Mostrou-se uma ferramenta de uma gama de opções que precisa ser estuda e entendida para realizar testes com detalhamento.

4.5 BUSCA DE LOGS DE SERVIDORES WEB APACHE

Os arquivos de logs dos servidores Apache reais foram utilizados em ambiente simulado como métodos de medição do tráfego da web, pois neles se encontram a maioria das informações necessárias para caracterizar a carga de trabalho.

Uma das vantagens em se trabalhar com esse tipo de medição é que não necessita em ocupar um servidor real para isso, o que poderia implicar no desempenho ou em bugs que poderia causar durante uso de ferramentas para capturar as informações do tráfego e implicaria numa maior proporção quando utilizadas ferramentas para gerar carga. Outra vantagem em trabalhar com ambiente simulado é que pode-se testar numa escala maior em um intervalo de tempo menor testando todo tipo de implicações que poderia ocorrer num ambiente real. Outra questão é que utiliza um número menor de dados para manipular sem perder as características das cargas reais.

A escolha em utilizar arquivos de logs de servidores Web, foi devido a não conseguir nenhuma empresa ou instituição que autorizasse a monitoração de servidores Web em seu ambiente de trabalho real, às mesmas usam políticas internas a qual não autorizam pessoas não ligadas às empresas a monitorar seus servidores Web.

A utilização dos logs como método de avaliação e monitoramento do tráfego da Web, foi optado por possuir trabalhos correlatos de autores como Martin Arlitt, Carey Williamson e Tai Jin, o qual foram marco nessa área de pesquisa, e que utilizaram dessa mesma técnica de medição.

Por meio de pesquisa na Internet, encontraram-se alguns sites que disponibilizam arquivo de logs de servidores em seu formato bruto (texto), para que outros pesquisadores nessa área tivessem acesso a esse material que quase sempre é restrito, e com isso pudessem fazer as extrações das principais características que compõe uma requisição HTTP.

E possivelmente utilizar esses levantamentos de métricas para desenvolvimento de softwares melhorados e também para construção de geradores de carga sintéticas para servidores Web, afim de testar seu desempenho, já que com as informações contidas nos arquivos de logs é possível construir um modelo de carga que mantém todas as características das cargas reais.

Antes de partir para as análises e caracterização da carga, devemos abordar os assuntos de técnicas de medição bem como os formatos dos logs do Servidor Web Apache.

4.5.1 Técnicas de Medição

As medições da Web podem ser coletadas de diversas maneiras. Os componentes de software, como *browsers, proxies* e servidores, podem gerar *logs*, como tratamento dos pedidos. Para tratar das características da carga de trabalho do servidor web, utiliza-se da técnica de monitoração pelo log do servidor.

4.5.1.1 Logging do Servidor

O servidor Web normalmente gera um *log* como parte do processamento dos pedidos do cliente. Cada entrada corresponde a um pedido HTTP tratado pelo servidor, tendo informações sobre o cliente que fez a solicitação, o horário e data do pedido, e mensagens de pedido e resposta. A padronização nos formatos dos *logs* facilitou o desenvolvimento de ferramentas comerciais para a análise. As estatísticas oferecem informações para diversos meios, como para administradores do site e de rede, desenvolvedores de conteúdo, pesquisadores para estudar os padrões de acesso de um grupo de clientes a um conjunto de recursos. Os logs dos servidores têm formado a base para a maioria dos estudos de pesquisa que têm caracterizado o tráfego HTTP ou avaliado novas técnicas para melhorar o desempenho da Web (KRISHNAMURTHY; REXFORD, 2001).

4.5.2 Formato do Log Apache

O log do servidor apache é disposto em modo textual, dessa forma permite ter um documento informando possíveis ocorrências que o servidor pode sofrer. É possível analisar as requisições, erros, sobrecargas, acesso a locais restritos entre outros.

A cada pedido do cliente ao servidor Web, informações sobre o procedimento de requisição e resposta são armazenados no *log.* A escolha em utilizar log do apache para estudo da caracterização da carga de trabalho, foi escolha devido às informações estarem disponibilizadas na Web para estudo.

O apache possui dois formatos de arquivos, o *Error Log* e o *Access Log*. O primeiro grava qualquer erro ocorrido no servidor.

Exemplo de uma mensagem de erro:

```
[Tue Jun 09 18:00:33 2009] [301] [client 209.175.180.194] cliente denied by server configuration: /home/htdocs/teste.
```

Entre os primeiros colchetes estão a data tempo de chegada. O segundo informa o código do erro [301], mostra o tipo de erro ocorrido. E o terceiro [cliente 209.175.180.194] informa o endereço IP do cliente que acessou o servidor. Logo após uma mensagem da navegação e o caminho do diretório ou arquivo não encontrado ou acessado.

O outro arquivo *Access Log*, armazena informações de todos os pedidos atendidos pelo servidor, incluído as requisições gravadas no *Error Log*. Esse arquivo possui dois tipos de formatos, o *Common Log Format* (comum) ou *Custom Log Format* (completo). No formato comum armazena informações básicas, e no completo, além das informações básicas, o tipo de navegador do cliente e o sistema operacional são gravados. O exemplo a seguir contém uma requisição no formato comum.

```
209.175.180.194 - - [17/May/2009:03:28:20 +0400] "GET /objeto.gif HTTP/1.1" 200 33628
```

O primeiro elemento identifica o endereço IP do cliente que gerou a requisição. O segundo e o terceiro campo informam o *identd* e o *userid* das máquinas dos clientes. Geralmente estes campos são em branco representados pelos dois hífens. Dos logs analisados somente o World Cup 1998 possuía essas informações. O *identf* por não ser confiável são mais utilizados em redes internas. O *userid* informa o usuário que requisitou a página, por meio da autenticação HTTP, identifica sessões dos clientes.

Entre os colchetes estão a data e a hora de chegada da requisição. O símbolo +0400 ao final, indica a localização do cliente.

Na seqüência após as aspas algumas das principais características utilizadas para a caracterização da carga de trabalho em um servidor Web. Representado sempre por letras maiúsculas, determina o método de acesso para solicitar um documento. Após o método, o caminho e o arquivo solicitado. O tipo de arquivo pode ser identificado pela sua extensão. Logo após a versão do protocolo utilizado para a comunicação é identificada ao final. Logo após, o código de resposta da requisição e o tamanho do arquivo em bytes finaliza uma linha de requisição (APACHE, 2009).

4.5.3 Identificação dos Logs Capturados

O objetivo desta sessão é demonstrar a identificação dos logs capturados na Internet para a realização das análises e da caracterização da carga de trabalho presente no estudo desse trabalho de pesquisa.

Os arquivos de logs eram disponibilizados em seu formato textual, ou seja, em seu formato bruto, e foram capturados de sites que disponibilizam esses para *download* para fins de pesquisa, de desenvolvimento ou de outras áreas interessadas.

Dos que foram analisados a maioria possuía o formato comum (*Common Log Format)*, mas disponibilizavam as informações necessárias para caracterizar a carga de trabalho. Nas sub-sessões abaixo será apresentado um conjunto de oito logs que foram coletados na Internet, sendo os da NASA, Saskatchewan, Calgary, ClarkNet, World Cup 98, e estão disponíveis no endereço: http://ita.ee.lbl.gov/html/traces.html. Já os demais como os da Stanford, Cdmil, e HVLP, foram disponibilizados nos próprios sites das respectivas instituições.

4.5.3.1 Log do Servidor do Kennedy Space Center

Esse log foi coletado entre o ano de 1994 e 1995, e suas requisições HTTP foram extraídas do servidor Kennedy Space Center da NASA, Flórida. Coletados por Jim Dumoulin.

```
d104.aa.net - - [01/Jul/1995:00:00:15 -0400] "GET /shuttle/countdown/count.gif HTTP/1.0" 200 40310
d104.aa.net - - [01/Jul/1995:00:00:15 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 200 786
d104.aa.net - - [01/Jul/1995:00:00:15 -0400] "GET /images/KSC-logosmall.gif HTTP/1.0" 200 1204
129.94.144.152 - - [01/Jul/1995:00:00:17 -0400] "GET /images/ksclogo-medium.gif HTTP/1.0" 304 0
199.120.110.21 - - [01/Jul/1995:00:00:17 -0400] "GET /images/launch-logo.gif HTTP/1.0" 200 1713
```

Figura 20. Exemplo log Kennedy Space Center

4.5.3.2 Log do Servidor da Universidade de Saskatchewan

Os logs foram extraídos do servidor HTTP da universidade de Saskatchewan, localizada em Saskatoon, Canadá, 1995. Coletados por Earl Fogel.

```
interchg.ubc.ca - - [01/Jun/1995:00:24:17 -0600] "GET /~lowey/encyclopedia/help.html" 200 2570
interchg.ubc.ca - - [01/Jun/1995:00:24:59 -0600] "GET /~lowey/encyclopedia/atlas.html" 200 691
interchg.ubc.ca - - [01/Jun/1995:00:25:16 -0600] "GET /~lowey/encyclopedia/m/maps.html" 200 1426
info.curtin.edu.au - - [01/Jun/1995:00:25:25 -0600] "GET /~scottp/publish.html" 200 271
yanafal.tele.nokia.fi - - [01/Jun/1995:00:26:31 -0600] "GET /~scottp/publish.html" 200 271
```

Figura 21. Exemplo log Saskatchewan

4.5.3.3 Log Servidor da Universidade de Calgary

Servidor do departamento de ciência da computação da universidade, localizada em Calgary, Canadá, 1995. Com contribuição de Robert Fridman.

```
local - - [24/Oct/1994:13:41:41 -0600] "GET index.html HTTP/1.0" 200 150
local - - [24/Oct/1994:13:41:41 -0600] "GET 1.gif HTTP/1.0" 200 1210
local - - [24/Oct/1994:13:43:13 -0600] "GET index.html HTTP/1.0" 200 3185
local - - [24/Oct/1994:13:43:14 -0600] "GET 2.gif HTTP/1.0" 200 2555
local - - [24/Oct/1994:13:43:15 -0600] "GET 3.gif HTTP/1.0" 200 36403
```

Figura 22. Exemplo log Calgary

4.5.3.4 Log Servidor da ClarkNet

É um provedor de acesso para o metrô de Baltimore - Washington, DC, 1995. Os registros foram coletados por Stephen Balbach.

```
shep102.wustl.edu - - [28/Aug/1995:00:01:05 -0400] "GET /pub/pribut/chess1.gif HTTP/1.0" 200 16254
shep102.wustl.edu - - [28/Aug/1995:00:01:06 -0400] "GET /pub/pribut/fancylin.gif HTTP/1.0" 200 2392
er6.rutgers.edu - - [28/Aug/1995:00:01:06 -0400] "GET /pub/rjgula/attacks.htm HTTP/1.0" 200 6950
ppp19.glas.apc.org - - [28/Aug/1995:00:01:07 -0400] "GET /atomicbk/new/new.html HTTP/1.0" 304 –
shep102.wustl.edu - - [28/Aug/1995:00:01:08 -0400] "GET /pub/pribut/new.gif HTTP/1.0" 200 147
```

Figura 23. Exemplo log ClarkNet

4.5.3.5 Log da World Cup 98

O log é do servidor HTTP da *World Cup 98* (Copa do Mundo de 1998), realizada na França. Este foi objeto de estudo de Martin Arlitt e Tai Jin , e em 1999 publicaram o artigo *Workload Characterization of the 1998 World Cup Web Site*.

A coleção desses foram divididas por dia de acesso e utilizou-se para diversos estudos sobre caracterização do tráfego da *Web*. O trace obteve mais de 1,3 bilhões de acessos durante um período de três meses. Os registros foram coletados de 33 servidores HTTP com quatro localizações geográficas diferentes: Paris (França); Plano (Texas); Herndon (Virgínia) e Santa Clara (Califórnia).

Por ser um log muito grande foi capturado informações do período de cinco dias, durante esse tempo ocorreu mais de quatro milhões de requisição.

```
46933 - - [03/May/1998:22:00:04 +0000] "GET /images/s102357.gif HTTP/1.0" 200 173
47558 - - [03/May/1998:22:00:04 +0000] "GET /english/nav_top_inet.html HTTP/1.0" 200 374
47558 - - [03/May/1998:22:00:04 +0000] "GET /english/nav_inet.html HTTP/1.0" 200 2672
47558 - - [03/May/1998:22:00:04 +0000] "GET /english/splash_inet.html HTTP/1.0" 200 3703
47524 - - [03/May/1998:22:00:04 +0000] "GET /english/frntpage.htm HTTP/1.0" 200 12512
```

Figura 24. Exemplo log World Cup 98

4.5.3.6 Log da Universidade de Stanford

O log da universidade de Stanford foi disponibilizado no endereço <http://waas.stanford.edu/~wwu/logs/access_log>, período de tempo disponibilizado foi entre os anos de 2002 e 2003.

```
163.28.48.69 - - [01/Feb/2002:00:17:41 -0800] "GET /figs/einstein.gif HTTP/1.0" 200 42750
163.28.48.69 - - [01/Feb/2002:00:17:42 -0800] "GET /figs/aa.logo.gif HTTP/1.0" 200 1326
163.28.48.69 - - [01/Feb/2002:00:17:42 -0800] "GET /figs/soe.logo.gif HTTP/1.0" 200 3594
163.28.48.69 - - [01/Feb/2002:00:17:43 -0800] "GET /figs/su.logo.gif HTTP/1.0" 200 947
163.28.48.69 - - [01/Feb/2002:00:17:43 -0800] "GET /figs/waassat.gif HTTP/1.0" 200 180116
```

Figura 25. Exemplo log Stanford

4.5.3.7 Log da Cdmil Webhosting

É uma empresa de estatísticas de webhosting, seu log está disponível no endereço <http://www.cdmil.com.br/logs/access_log>, coleta realizada em 2009. Nota-se no exemplo da Figura 26 que este encontra-se no formato completo podendo extrair outras informações como *browser* do cliente e sistema operacional.

```
70.84.81.18 - - [24/May/2009:04:18:01 -0300] "GET /cgi-bin/Count.cgi?sh=1 HTTP/1.0" 200 362 "-"
"Lynx/2.8.5dev.7 libwww-FM/2.14 SSL-MM/1.4.1 OpenSSL/0.9.7a"
65.55.208.159 - - [24/May/2009:04:21:06 -0300] "GET /robots.txt HTTP/1.1" 200 383 "-" "msnbot/1.1
(+http://search.msn.com/msnbot.htm)"
65.55.208.159 - - [24/May/2009:04:21:08 -0300] "GET /camera.php HTTP/1.1" 200 77738 "-" "msnbot/1.1
(+http://search.msn.com/msnbot.htm)"
```

Figura 26. Exemplo log Cdmil Webhosting

4.5.3.8 Log da HVLP

Também é uma empresa, site tradicional de informação, seu log está disponível no endereço <http://www.hvlp.com/logs/access_log>, coleta realizada em 2009.

```
65.55.106.115 - - [17/May/2009:04:22:03 +0400] "GET /robots.txt HTTP/1.0" 404 33628 "-" "msnbot/2.0b
(+http://search.msn.com/msnbot.htm)"
65.55.106.115 - - [17/May/2009:04:23:04 +0400] "GET /private/server/logs/?M=D HTTP/1.0" 200 7234 "-"
"msnbot/2.0b (+http://search.msn.com/msnbot.htm)"
220.181.7.97 - - [17/May/2009:05:08:57 +0400] "GET / HTTP/1.0" 200 871 "-"
"Baiduspider+(+http://www.baidu.com/search/spider.htm)"
```

Figura 27. Exemplo log HVLP

4.6 ETAPAS DA CARACTERIZAÇÃO DA CARGA DE TRABALHO

O objetivo é demonstrar as etapas da caracterização, como caracterizar a carga já em logs capturados e resultados obtidos. Para isso utilizando os softwares descritos na sessão 4.2 e aplicando a metodologia para caracterização da carga de trabalho.

4.6.1 Informações Gerais

Os arquivos de *logs* são compostos de dados com informação sobre o pedido de uma requisição pelo usuário e da resposta do servidor *Web*. Para o estudo da caracterização da carga de trabalho coletou-se um conjunto de oito *logs* de servidores Web Apache.
Os *traces* analisados estavam todos disponibilizados na Internet e a partir dos dados brutos foi feito o pré-processamento.

Eles possuem diferentes magnitudes, desde mais de cinco milhões de requisições, World Cup 98, até poucas requisições como o da HVLP, com cerca de cinco mil. As datas de coleta de requisições são distintas, exemplo o da NASA, Sashatchewan, Calgary, Clarknet são do ano de 1995, da Stanford 2002 e 2003 e os da Cdmil e HVLP são de 2009. O período de análise varia de cinco dias (Word Cup 98), a um ano (Stanford).

Tabela 10. Descrição dos Logs Pré-Processados

logs analisados	Quantidade de requisições	Período de análise (início-fim)
Calgary	725.098	24/10/94 - 11/10/95
Saskatchewan	445.475	01/06/95 - 31/07/95
Kennedy Space Center (NASA)	717.412	01/07/95 - 10/07/95
Clarknet	1.654.930	28/08/95 - 03/09/95
World Cup 98	5.088.464	01/05/98 - 05/05/98
Stanford	920.246	01/02/02 - 04/02/03
HVLP	5.183	17/05/09 - 02/06/09
Cdmil	184.676	24/05/09 - 02/06/09

4.6.2 Dados Pós Processamento

Após o processamento eliminou-se as linhas inválidas, ou seja, as que não possuíam todos os dados para uma possível análise, como métodos de acesso vazio ou que não tinham o código de resposta do servidor, entre outras informações que completa uma linha de requisição. Na Tabela 11 observa-se que o *trace* da Calgary (1995) possui uma porcentagem 0,04% de linhas eliminadas, já a Stanford (2003) apresentou 10,87 %, esse valor significa que mais de 100 mil eram linhas inválidas. Os bytes transferidos vão de 64 Gigabytes para a Stanford (2003) e 92 Megabytes para HVLP (2009).

Tabela 11. Quantidade de requisições pós processamento, porcentagem das linhas eliminadas e GBytes transferidos

logs analisados	Quantidade de requisições pós processamento	% linhas eliminadas	GBytes transferidos
Calgary	724.837	0,04	7,40
Saskatchewan	416.533	6,50	2,25
Kennedy Space center (NASA)	714.776	0,37	15,14
Clarknet	1.652.159	0,17	13,43
World Cup 98	4.738.745	6,87	38,86
Stanford	820.183	10,87	64,87
HVLP	5.161	0,42	0,09
Cdmil	169.927	7,99	18,33

Informações sobre a quantidade de requisições por dia e hora são importantes para estabelecer se o servidor está atendendo a demanda, assim é possível fazer um planejamento quanto aos recursos de máquina para que haja uma satisfação em tempo de resposta do servidor *Web*.

Tabela 12. Descrição da quantidade de requisições realizadas por dia e hora

logs analisados	Média de requisições por dia	Média de requisições por hora	Média de Bytes transferidos por dia	Média de Bytes transferidos por hora
Calgary	2.053	85,56	22.509.356	937.889
Saskatchewan	6.828	284,52	39.560.174	1.648.341
Kennedy Space Center (NASA)	71.478	2.978,23	1.626.075.913	67.753.163
Clarknet	236.022	9.834,27	2.063.286.273	85.970.261
World Cup 98	947.749	39.941,19	8.345.463.005	351.439.707
Stanford	2.222	92,78	188.773.053	7.879.780
HVLP	303	12,74	5.726.341	240.365
Cdmil	16.992	718,50	1.968.661.706	83.296.164

4.6.3 Características Gerais dos Logs

O Conjunto dos oito *logs* analisados foram utilizados para o estudo de caracterização da carga de trabalho de um servidor *Web*, o servidor em questão que será utilizado é o Apache, desse é que foram coletados os registros brutos, que se encontra em modo textual.

Do conjunto é possível analisar o método de acesso para requisitar um documento, versão do protocolo HTTP utilizados pelos clientes Web, código resposta do servidor, classe de objeto, tamanho do arquivo, intervalo de chegada dos pedidos, entre outros dados que serão analisados a partir dos *traces*.

4.6.4 Métodos de Acesso

Entre os tipos de métodos de acesso que podem ser utilizados para solicitar um documento ao servidor, estão: o GET solicita que seja retornado o recurso identificado pela URL. Esse é o mais utilizado entre os métodos de acesso, pois é empregado por padrão pelos *user-agent*. O método incluí dois tipos, o GET condicional, requisições com cabeçalho *if-Modified-Since*, e o GET parcial, com o cabeçalho *Range.* No entanto não é possível, determinar essas informações, usando somente o *log* como análise, o que seria uma informação muito importante na análise do impacto dos clientes e *proxy* na carga de trabalho do servidor.

Outras implicações que o torna o mais comum é que possui a propriedade de ser *safe* (seguro), ou seja, não deve ser utilizado para modificar dados que estão no servidor. Por exemplo, atualizar um dado em um banco de dados, e a outro atributo importante é *idempotent* (idempotente), essa significa que múltiplas requisições ao mesmo recurso usando o método, devem ter o mesmo resultado de uma única requisição.

Além do GET outros métodos que também são utilizados numa requisição HTTP são: HEAD, POST, PUT, OPTIONS, CONNECT, TRACE e DELETE.

O HEAD obtém informação sobre o recurso sem que o mesmo seja retornado ao cliente, esse também partilha das propriedades *seguro* e *idempotente*. Concluindo a seqüência, o POST envia informações adicionais do cliente para o servidor no corpo da mensagem. O PUT permite criar e modificar um recurso no servidor Web. O OPTIONS permite determinar as opções e requisitos associados aos recursos solicitados, sem necessariamente iniciar sua recuperação. O CONNECT é reservado para comunicação do servidor *proxy*. O TRACE é usado para enviar uma mensagem de teste ao servidor do tipo *loopback*. Já o DELETE, solicita que o servidor Web remova o objeto definido pela URL. Além do GET e HEAD,

métodos como: PUT e DELETE também são *idempotentes*, assim como o OPTIONS e TRACE, que não tem efeitos colaterais, por isso são inerentemente idempotente.

Tabela 13. Porcentagem dos métodos de acesso utilizado pelo protocolo HTTP

logs analisados	GET	HEAD	POST	PUT	OPTIONS	CONNECT
Calgary	99,90	0,08	0,02	0,00	0,00	0,00
Saskatchewan	99,96	0,00	0,04	0,00	0,00	0,00
Kennedy Space Center (NASA)	99,83	0,17	0,00	0,00	0,00	0,00
Clarknet	99,65	0,13	0,22	0,00	0,00	0,00
World Cup 98	99,62	0,08	0,30	0,00	0,00	0,00
Stanford	99,89	0,01	0,07	0,00	0,03	0,00
HVLP	99,98	0,00	0,02	0,00	0,00	0,00
Cdmil	99,96	0,00	0,04	0,00	0,00	0,00
Média	99,85	0,06	0,09	0,00	0,00	0,00
Mediana	99,89	0,04	0,04	0,00	0,00	0,00
Desvio Padrão	0,14	0,07	0,11	0,00	0,01	0,00

O método GET por ser mais comum obteve uma mediana de 99,89% dos pedidos válidos, quase a totalidade. Os métodos HEAD, POST obtiveram a mediana 0,04% os demais PUT, OPTIONS e CONNECT tiveram valores desconsiderados, já o TRACE e DELETE não foram encontrados nos *logs* analisados.

4.6.5 Protocolos HTTP

A versão dos protocolos HTTP também é possível de serem analisados por meio dos *logs* de acesso do servidor *Web*. Duas versões de protocolos foram detectadas a HTTP/1.0, essa possui uma série de conexões para cada Requisição/Resposta e a versão HTTP/1.1 possui várias sessões ativas utilizando uma única conexão.

Tabela 14. Resumo das versões do protocolo HTTP

logs analisados	Quantidade de requisições HTTP/1.0	%	Quantidade de requisições HTTP/1.1	%
Calgary	724.837	100	0	0,00
Saskatchewan	416.533	100	0	
Kennedy Space center (NASA)	714.776	100	0	0,00
Clarknet	1.652.136	100	23	0,00
World Cup 98	3.721.365	78,53	1.017.380	21,47
Stanford	359.260	43,80	460.923	56,20
HVLP	6	0,12	5.155	99,88
Cdmil	10.723	6,31	159.204	93,69

Observa-se que os quatro primeiros *logs* descritos na Tabela 14, que as versões dos protocolos utilizados eram de 100% HTTP/1.0, isso no ano de 1995, que era bem atual para a época, já que os primeiros rascunhos surgiram em meados de 1993.

Já o log da Stanford (2003) demonstra que houve um aumento na utilização do HTTP/1.1, que obteve 56,20% superando em 12,40% a versão 1.0, isso devido à utilização de navegadores mais atualizados. Nos mais atuais, como o da Cdmil (2009) e HVLP (2009), o HTTP/1.1 tem mediana de 96,79%, se tornando o padrão para a atualidade.

Na Figura 28 pode-se visualizar que em uma década houve um decréscimo no emprego do protocolo HTTP 1.0 e um aumento no HTTP1.1.

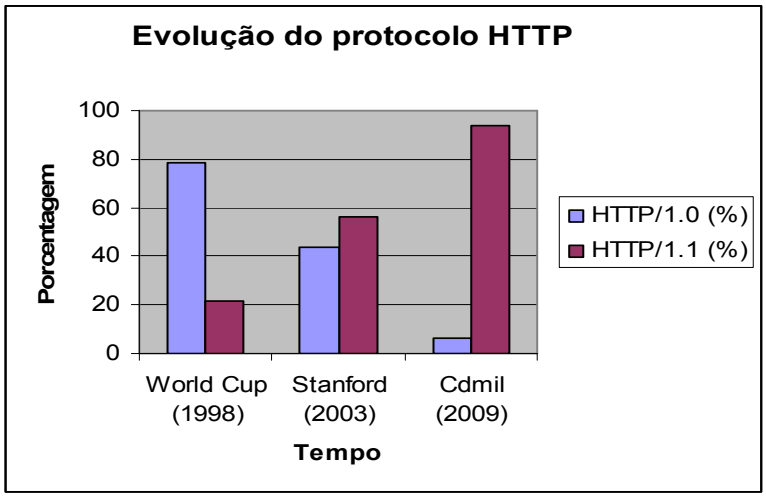

Figura 28. Evolução do protocolo HTTP

4.6.6 Código Resposta

Ao receber uma requisição, o servidor procura pelo recurso e envia uma resposta ao cliente. Essa contém um código de resposta que informam ao cliente o *status* ao atender a requisição.

Os códigos resposta (*status*) foram separados em quatro classes distintas: A primeira 2xx, indica que teve como resposta *successful,* ou seja, requisição recebida, entendida e processada com sucesso. Na 3xx redireciona o cliente para outra URL, isso é feito pelo próprio *browser*. A classe 4xx indica que há erros na requisição do cliente e a 5xx indica a incapacidade do servidor em processar o pedido de requisição e também pode indicar uma sobre carga no servidor.

Tabela 15. Descrição dos códigos respostas do protocolo HTTP

Código de Resposta	Média %	Mediana	Desvio Padrão
2xx	86,27	92,31	19,45
3xx	4,93	1,12	6,35
4xx	8,79	1,38	19,93
5xx	0,01	0,01	0,01

Na Tabela 15, observa-se que os códigos de resposta utilizando a classe 2xx possui uma média de 86,27% das requisições válidas, indicando que os pedidos tiveram *SuccessfulI*, ou seja, sucesso como resposta. Utilizando a classe 3xx ficou com média 4,93% indicando um redirecionamento para completar o pedido. A classe 4xx com a média de 8,79% superou a classe 3xx, essa classe indica que houve erro na requisição do cliente, como por exemplo, acesso a uma região que não existe no servidor. O aumento significativo desta classe deve-se ao *trace* da HVLP (2009) que obteve média de 57,90% da classe 4xx, nota-se que o desvio padrão para essa classe ficou bem alto com o valor de 19,93% das requisições válidas, superando inclusive os da 2xx que tiveram média de 41,40%. E por fim a classe 5xx

teve mediana de 0,01%, indicando a alta disponibilidade dos servidores Web Apache, já que essa classe indica erro no servidor.

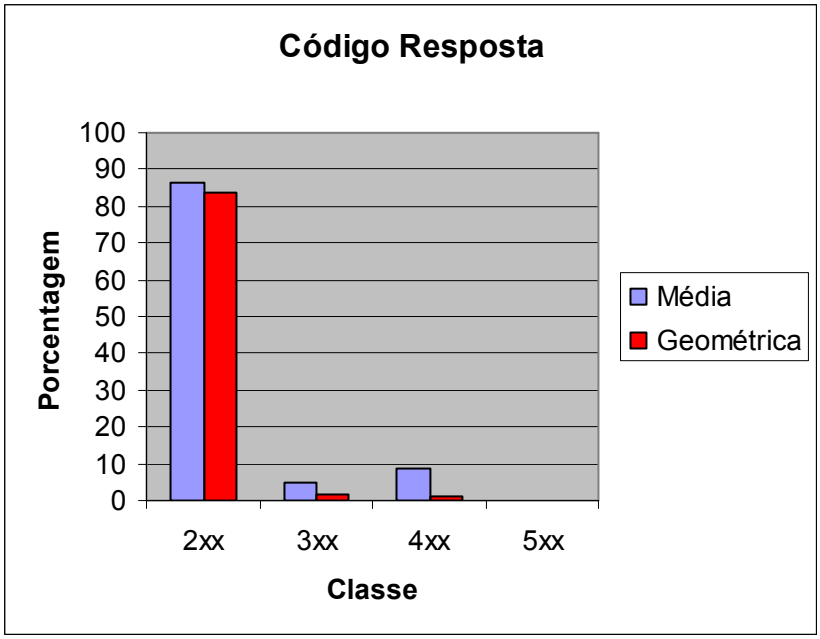

Figura 29. Códigos de resposta

A Figura 29 mostra a média de requisições por classe de código de resposta e a distribuição geométrica. Essa distribuição é utilizada por algumas ferramentas que geram cargas de trabalho sintéticas para melhor representar os resultados obtidos.

O cruzamento de dados é muito importante para uma análise mais detalhada de um item em questão, sendo assim, para quantificar métodos de acesso que tenham código de resposta 200 (*successfull*), usou-se o cruzamento de dados.

Tabela 16. Porcentagem de requisições por método de acesso utilizando código de resposta 200

Método	Média %	Mediana	Desvio Padrão	Média de Bytes Transferidos (%)
GET	99,87	99,93	0,15	99,97
POST	0,04	0,00	0,08	0,02
HEAD	0,06	0,04	0,07	0,01

Observando o resumo da Tabela 16, nota-se que a porcentagem de requisição utilizando o método GET, tendo como código de resposta 200 ficou em torno de 99,87 %, assim como os bytes transferidos, que tiveram uma média de 99,97% do cruzamento de dados. Interpretando os resultados pode-se dizer que o servidor Web apache, tem uma grande eficiência em atender as requisições, mesmo com uma grande demanda de pedidos HTTP.

Uma descrição dos códigos de resposta do servidor Web para o conjunto dos oito *logs* analisados está representada pela Tabela 17.

Tabela 17. Porcentagem de código resposta HTTP por descrição

Código de Resposta	Média (%)	Mediana	Desvio Padrão	Média de Bytes Transferidos (%)
200	83,88	89,57	18,82	84,61
206	1,79	0,03	4,93	3,45
301	0,34	0	0,48	0
302	1,26	3,94	1,65	0,01
304	3,94	2,03	4,98	0
400	0,01	0	0,03	0
401	0	0	0	0
403	0,73	0,05	1,55	0,01
404	8,04	1,34	18,39	11,91
405	0	0	0	0
500	0,01	0	0,01	0
501	0	0	0	0

Alguns códigos de resposta serão descritos, de forma breve, para um melhor entendimento:

a) 200 *OK* - requisição atendida com sucesso. Junto com o *status* o servidor deve enviar, acompanhamento de alguns *headers*, o conteúdo do recurso requisitado que pode ser, por exemplo um documento HTML, uma imagem JPEG. No conjunto dos *logs* analisados, 83,88% das requisições teve esse código como resposta do servidor. Nota-se um desvio padrão alto com valor de 18,82. Por isso, o valor da mediana com 89,57% torna-se um valor mais aproximado;

b) 302 *Found* – este é código de redirecionamento. Descreve como sendo temporário, ou seja, pode ser que na próxima requisição não seja mais necessário um redirecionamento. Ao receber um código 302, o cliente deve procurar pelo *header "Location"*, que deve informar a URI para qual recurso está sendo redirecionado;

c) 304 *Not Modified* – é usado quando o cliente faz uso de *caching*, ou seja, guarda cópias locais dos recursos acessados. Informa ao cliente que o recurso não foi modificado desde a última requisição e que a versão guardada em cache pode ser usada. Esse código teve 3,94%, sendo o mais utilizado na classe 3xx;

d) 404 *Not Found* – informa ao cliente que o recurso não foi encontrado no servidor. Esse código de resposta teve um grande percentual com 8,04% do total dos oito *logs* analisados, superando inclusive o da classe 3xx. Isto foi devido ao *trace* da HVLP que teve um retorno de 53,00% de reposta com código 404. Isso é notado pelo alto valor do desvio padrão com 18,39. Então o ideal para esse caso seria representar esse valor pela mediana que obteve 1,34%;

e) 503 *Service Unavailable* – o recurso está temporariamente indisponível. Este erro pode ser causado por sobrecarga no servidor ou por alguma operação de manutenção. A monitoração desse tipo de erro pode ser muito útil, pois pode indicar que o servidor não está atendendo a demanda de requisições. Dos *logs* analisados o percentual desse tipo de erros foi desconsiderado, isto pode caracterizar que o servidor Web apache possui uma grande disponibilidade em atender as requisições dos clientes.

4.6.7 Classe de Objeto

A classe de objetos pela diversidade encontrada na Internet possui um grande impacto na caracterização da carga de trabalho e na avaliação de desempenho de servidores *Web*. Alguns autores e ferramentas de análises utilizam as extensões dos arquivos para classificar os objetos por classes, sendo assim, facilita as análises dos dados, já que cada linha de requisição possui um tipo de objeto definido pela extensão do arquivo.

Tabela 18. Classificação das classes dos objetos.

Classe dos Objetos	Sigla	Extensões
Imagem	Img	bdf, bmp, gif, ico, jpe, jpeg, jpg, pbm, png, tif, tiff e xbm.
Linguagem de Marcação	Htm	htm, html, shtml, wml e xml.
Dinâmico	Din	asp, aspx, cfm, cgi, cgi-bin, jsp, php, php3 e pl.
Texto	Tex	asc, bib, css, dat, idx, inf, java, log, map, ram, smi, tex, txt.
Documento	Doc	doc, dvi, eps, fdr, hfo, mso, pdf, pps, ppt, ps, pub, rtf, vsd, wbk, wpd, wri, xls.
Script, Cliente e Animação	Src	class, dcr, js, swf e vbs.
Áudio	Aud	aif, asf, au, mid, mp3, mp4 e wav.
Binário e Compacto	Bin	arj, bin, cab, cmd, db, dll, emz, exe, gz, jar, rar, rpm, tar, tgz, wmf, wmz e zip.
Vídeo	Vid	avi, mov, mpe, mpeg e wmv.
Outros	Out	tipo de extensão não classificada.

Os objetos serão definidos pelas classes como descrito na Tabela 18. Isso facilita a visualização, já que os objetos foram separados por siglas. Depois de classificados os objetos, chegou-se ao resultado descrito na Tabela 19.

Tabela 19. Resumo classe de objetos.

Classes de Objetos	Média de requisições (%)	Média de Bytes Transferidos (%)
Img	58,15	21,82
Htm	18,69	15,76
Din	13,82	22,78
Tex	3,72	1,03
Doc	3,93	15,89
Src	0,78	0,62
Aud	0,13	2,03
Bin	0,74	18,92
Vid	0,05	2,04
Out	0,00	0,38
Total	100	100

As classes de objetos com maior incidência são os de imagem com uma média 58,15% . Já na classe de linguagem de marcação as médias ficaram em torno de 18,69%. Na classe dinâmicas a média aproximou-se de 14%, este aumento deve-se ao *trace* da Cdmil (2009), que obteve 40,64% de objetos dessa classe. Isto demonstra que por ser um *log* bem recente houve um aumento significativo, devido ao uso de *scripts* executando no lado do servidor, como ASP, JSP e PHP. As demais classes somaram uma média de 9,00% das requisições.

Nota-se pelo gráfico da Figura 30, que mesmo objetos com poucas requisições como as classes dos binários 0,74% possui um grande número de bytes transferidos aproximando dos 19,00%, indicando que esses objetos podem ser suficientes para produzir sobrecargas na rede e no servidor. As classes dos dinâmicos obtiveram 22,00% das transferências e documentos tiveram 15,00%, assim como a linguagem de marcação que também obteve o mesmo percentual.

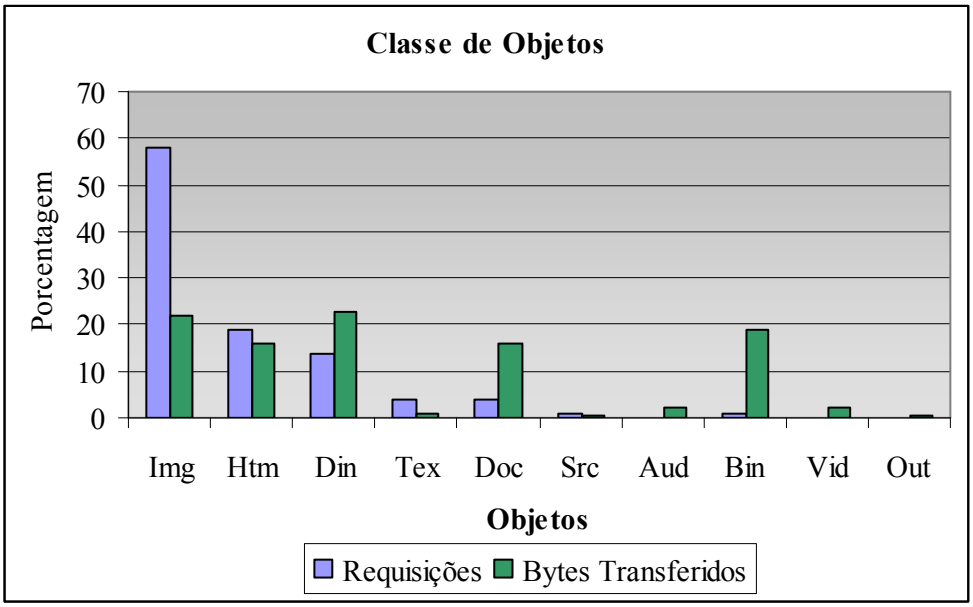

Figura 30. Resumo de requisições e bytes transferidos

Observa-se que os objetos do tipo imagem, não possuem um volume muito grande de bytes se comparando com o número de requisições, esse que ficou em torno de 21%. Os Scripts Clientes e Animação são objetos executados na máquina cliente, ou seja, no browser do usuário. Essa classe de objeto obteve 0,62% dos bytes transferidos. Outras classes de objetos como vídeo, imagem e texto obtiveram um pequeno número em volume de bytes transferidos não interferindo na carga de rede consecutivamente na do servidor.

4.6.8 Intervalo de chegada

O intervalo de chegada é diferença de tempo entre duas requisições sucessivas feitas ao servidor. Nesta etapa analisaram-se as quantidades de requisições em um determinado intervalo de tempo, e se chegou ao resultado como descrito na Tabela 20.

Tabela 20. Intervalo de tempo de chegada

Intervalo de Tempo	Quantidade de Requisições	%
<1s	3.227.326	55,28
1-10s	2.000.107	34,26
11-30s	296.939	5,09
30-60s	119.142	2,04
60- 300s	142.716	2,44
300-1800s	48.534	0,83
1800-3600s	2.929	0,05
3600s +	773	0,01
Total	5.838.466	100

Observa-se que há um grande número de requisição no intervalo de tempo inferior a 1 segundo (s), chegando a 55,28% das requisições, ou seja mais da metade são geradas neste intervalo de tempo, já a outra maior parte das requisições concentra-se no intervalo de 1s à 10s, com 34,26% . Logo nota-se que mais de 89,00% das requisições tem um tempo de chegada de menos de 1s à 10s. Infelizmente só com os logs não é possível calcular o tempo de resposta do servidor o que seria de extrema importância para o estudo de caracterização da carga de trabalho do servidor Web.

Para uma melhor avaliação, foram analisados os tipos de classes de objetos por intervalo de tempo de chegada, ou seja, onde ocorreu uma maior incidência de um determinado objeto e qual o intervalo tempo de chegada que tem maior porcentagem para esse objeto. Os dados foram dispostos do seguinte modo: as classes foram classificadas conforme a Tabela 18 e o intervalo de tempo conforme a Tabela 20.

Os objetos foram extraídos da seguinte maneira: imagem, linguagem de marcação, dinâmicos e texto foram coletados do *log* da Cdmil (2009). Compacto e binário, vídeo e áudio, do World Cup 98 (1998), já a classe dos documentos foram representados pelo *log* da Stanford (2003), as classes foram representadas pelos logs onde ocorreu um número

significativo dessas classes de objetos, por isso a escolha para melhor representar o intervalo de chegada e latência que pode ou não causar ao servidor.

Os objetos do tipo imagens ocorreram 55,94% no intervalo de chegada menor que 1 segundos (s), e seus tamanhos não ultrapassaram o intervalo de [0 Kb – 1000 Kb].

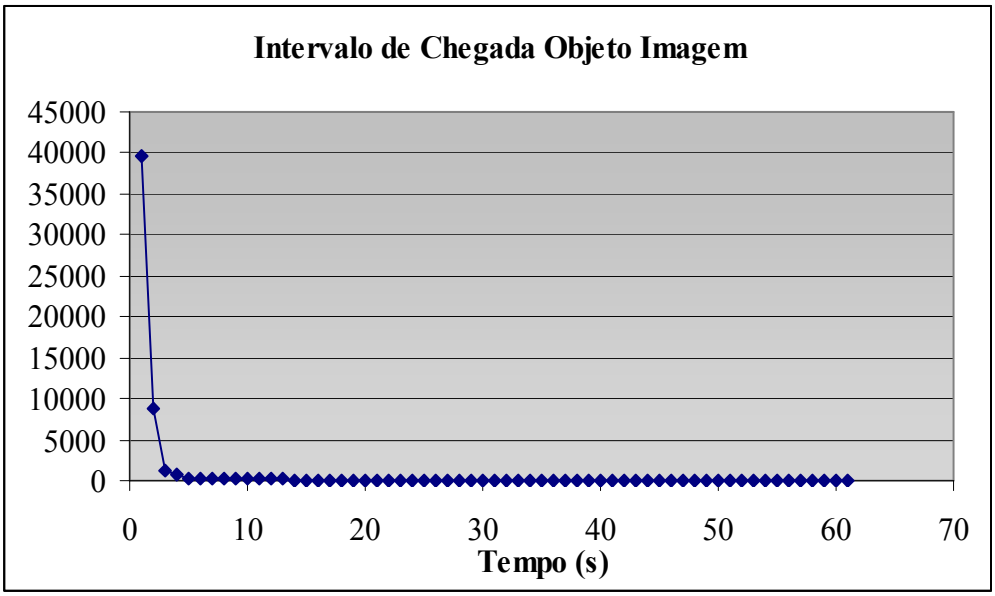

Figura 31. Intervalo de chegada objeto imagem

Os objetos de linguagem de marcação ocorrem na sua maioria com 41,33% no intervalo de [60s – 300s], já o tamanho desses objetos tiveram não ultrapassado 1000 Kb.

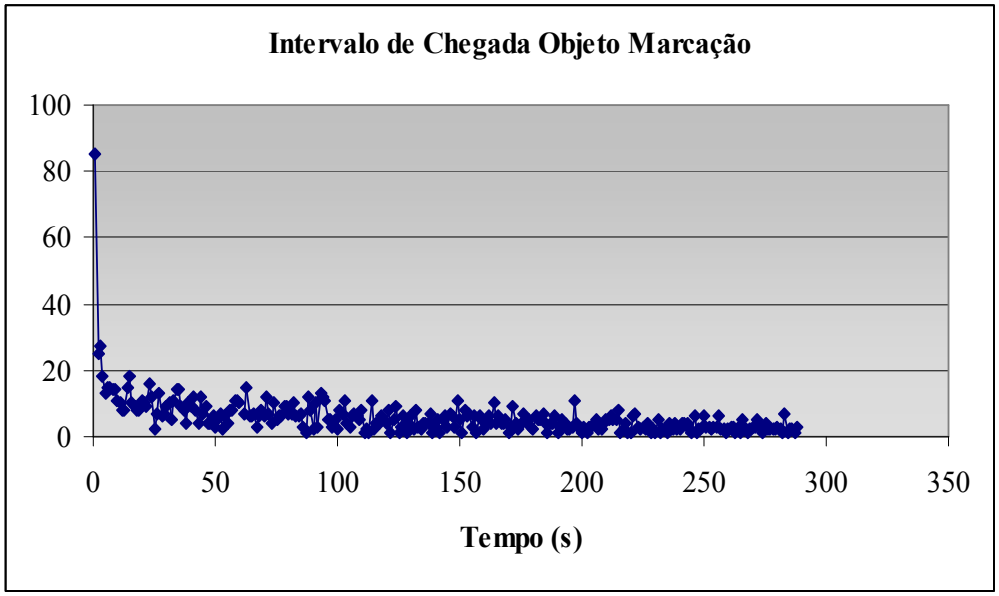

Figura 32. Intervalo de chegada objeto de marcação

Os objetos de classe dinâmicos são *scripts* executados junto ao servidor Web, esse tipo de objeto foi observado com uma maior porcentagem no período de tempo de [1s – 10s] com 49,67%. Esse tipo de objeto pode exigir mais dos processadores e podem sobrecarregar o servidor, esses que não ultrapassaram o intervalo de tamanho [1 – 1000 Kb].

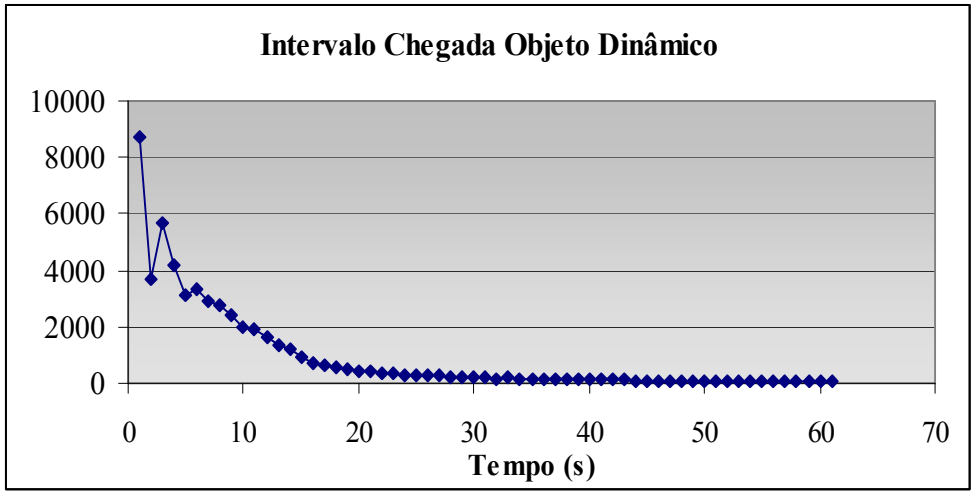

Figura 33. Intervalo chegada Objeto Scripts

Os objetos do tipo texto variaram bastante quanto ao tempo de chegada entre uma requisição e outra, o intervalo de 0s obteve 27,78%, já o intervalo de [1s – 10s] 21,80%, que

ficou também divido no intervalo de [60s – 300s] com 22,58%, variando bastante o intervalo de chegada. Já o tamanho deste objeto não ultrapassou o intervalo de [0 – 1000 Kb].

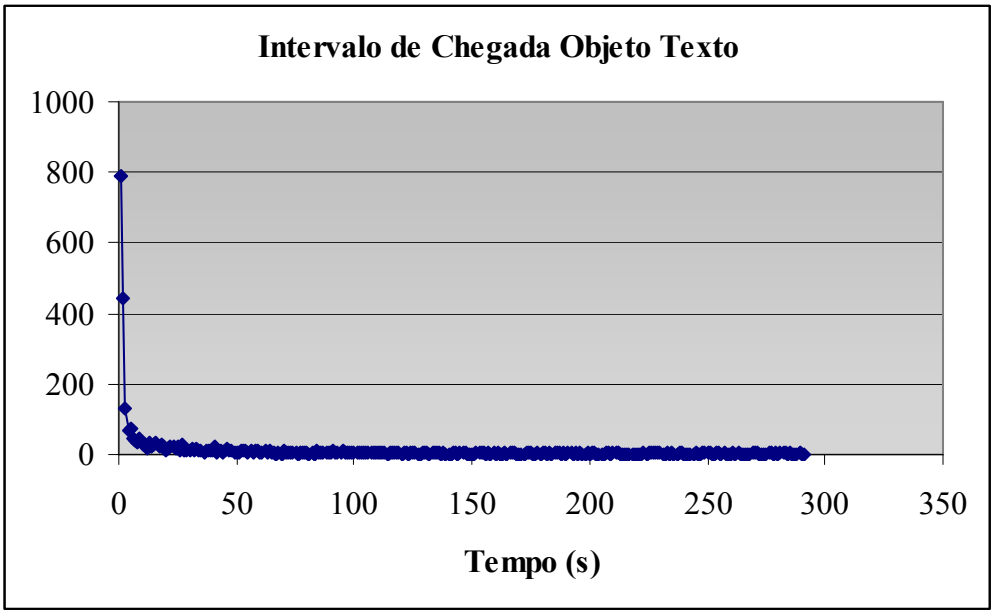

Figura 34. Intervalo de chegada objeto texto

Os scripts clientes e animação, concentraram-se em intervalos bem variados de [300s – 1800s], com uma média de 50,75% para este intervalo.

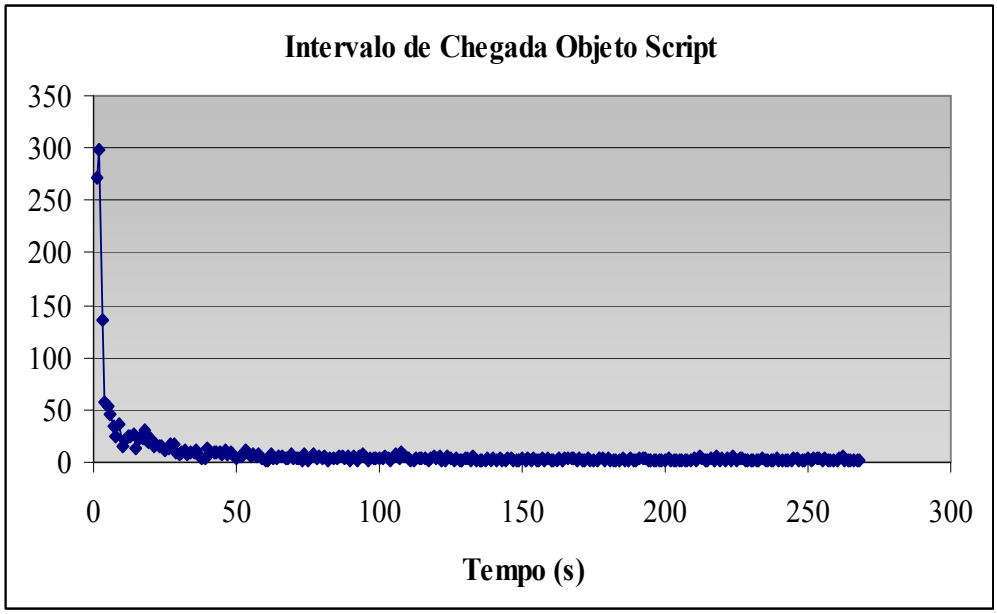

Figura 35. Intervalo de chegada Objeto Script

Os objetos compactos e binários ficaram em dois intervalos com sua maioridade, os tempos de chegada de [1s – 10s], com 38,86% e de [10s – 30s], com 31,89%. O tamanho desse objeto é bastante grande e ficou em torno de 1.500 Kb, por essa razão mesmo com poucas requisições para esse tipo de objeto, caracteriza que há um grande volume de dados envolvidos, podendo tornar a rede e o servidor sobrecarregados.

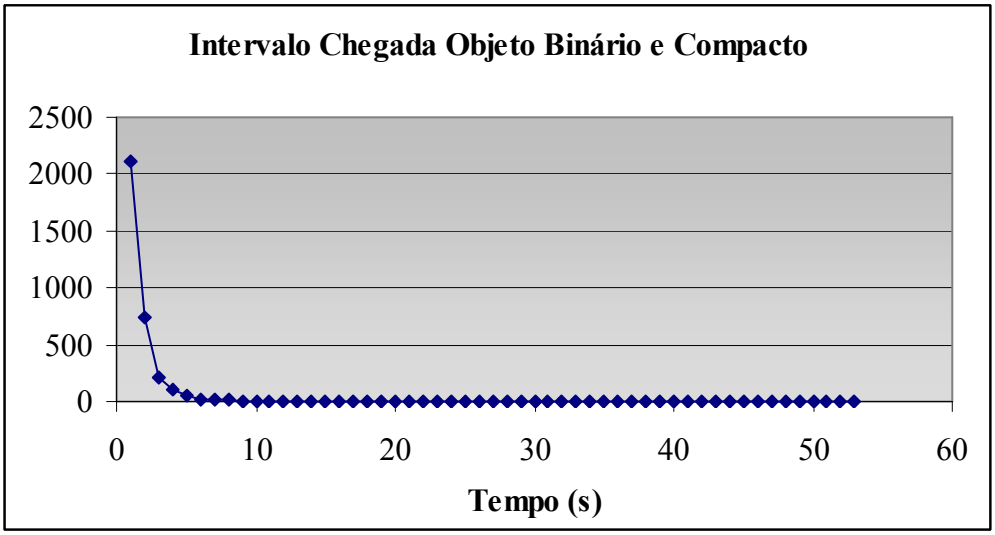

Figura 36. Intervalo chegada objeto binário e compacto

O mesmo ocorre com objetos da classe de vídeos, que tiveram o tamanho médio aproximado de 1.500 Kb. Esse ficou concentrado no tempo de chegada de [<1 – 10s] e [10 – 30s] com 37,12%.

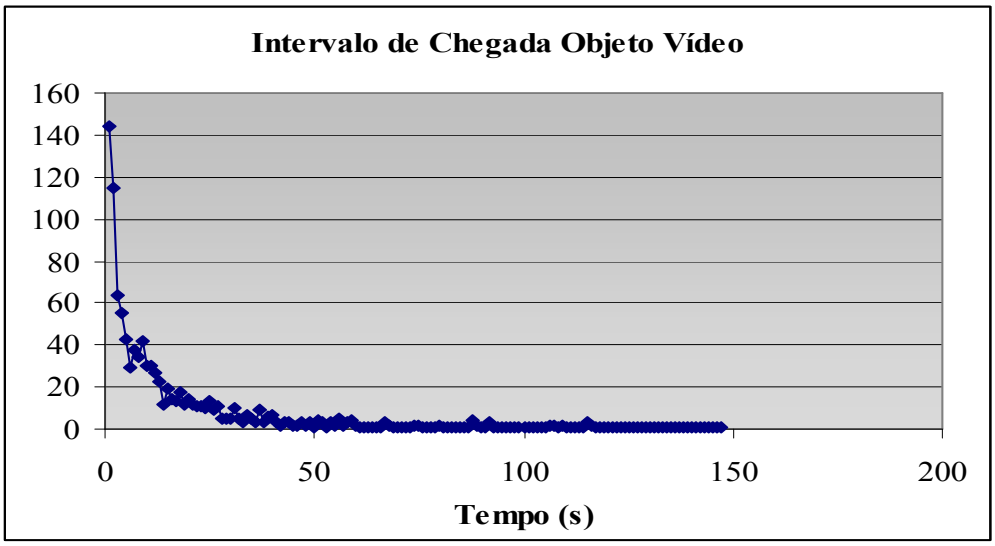

Figura 37. Intervalo de chegada objeto vídeo

Os objetos de áudio tiveram como intervalo [1- 10s] e [30s – 60s] com 37,50%, e seu tamanho em torno de 500 Kb.

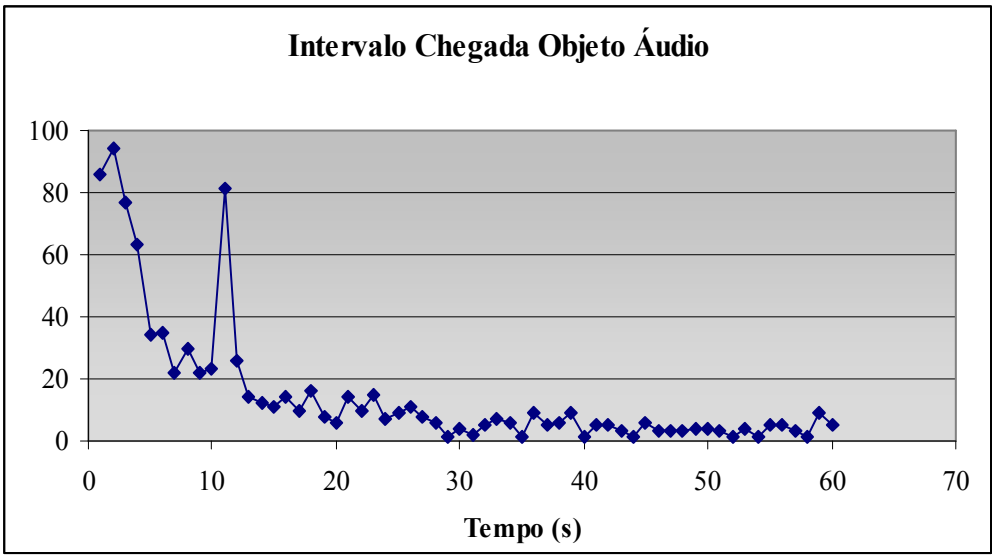

Figura 38. Intervalo de chegada objeto áudio

E por fim objetos tipo documentos que teve o intervalo de chegada entre uma requisição e outra o tempo de [1s – 10s], com 46,76%.

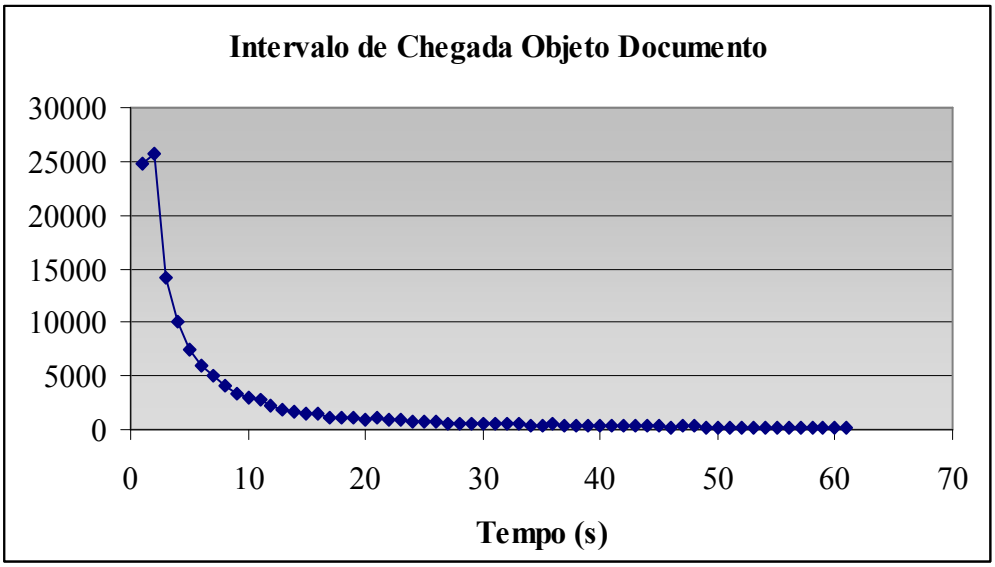

Figura 39. Intervalo de chegada objeto documento

4.6.9 Discussões e Considerações Finais

Diversas características compõem a carga de trabalho de um servidor Web, o estudo do conjunto de oito *logs* com diferentes magnitudes e diferentes épocas, pode-se observar a evolução da Web, como por exemplo, do protocolo HTTP/1.1, ou o aumento significativo do uso de scripts PHP, JSP junto ao servidor, que passaram de 0,4% em 1995 para 40% em 2009. Esse tipo de evolução contínua também pode afetar a carga de trabalho de um servidor. Já que objetos requisitados como vídeo, dinâmicos, binários mesmo quando pouco requisitados podem envolver um volume grande de dados e sobrecarregar o tráfego da Web.

Outro ponto de análise foram os métodos de acesso para um pedido de requisição pelo cliente e código resposta do servidor demonstrando a alta disponibilidade do servidor em atender a demanda de pedidos.

A principal ênfase das análises devem-se a utilização de ferramentas de leitura e de geradores de cargas. Com essas foi possível o levantamento dos dados necessários para

esse estudo, sem ter que utilizar um servidor real, já que por meio dos logs capturados é possível fazer essas medições, e usar um ambiente simulado para isso. O uso de um servidor real para a análise poderá implicar em alguns pontos negativos, como por exemplo, perda de desempenho já que terá que rodar ferramentas com o servidor em uso, e este problema pode ser mais amplo quando utilizadas softwares de benchmarks, em que se gera uma carga sintética em maior escala. Portanto, este tipo de análise torna-se mais prática em ambiente simulado, que ficou evidente nas etapas de medição para esse trabalho.

O estudo e levantamento das ferramentas foi uma questão bem discutida e trabalhosa para o desenvolvimento desse trabalho. Então por isso vale ressaltar algumas das características dessas ferramentas.

A ferramenta Awstats se mostrou uma ferramenta de fácil configuração e muito eficaz em capturar informações dos logs, retornando todo o resultado em gráficos de fácil visualização e entendimento. Alguns pontos fortes dessa ferramenta, é que consegue capturar os tipos de objetos requisitados, assim como navegador e sistema operacional utilizados pelos clientes, desde que o formato do log seja completo. Contra, não captura métodos de acesso, código de resposta e versão do protocolo HTTP.

A ferramenta Webalizer também se mostrou uma ferramenta muito rápida, e que captura a maioria das informações básicas, como quantidade de acesso, páginas visitadas, gera tudo graficamente. Além disso, consegue capturar os códigos de resposta do servidor. Contra, não captura os objetos, métodos de acessos, versão do protocolo.

Já a ferramenta AnalisadorlogsApache, foi a ferramenta de uma grande utilidade. A mesma é de uso simples, desenvolvida em C, e captura além das informações básicas, outras como: objetos requisitados, método de acesso, código de resposta e versão do protocolo. E o mais importante é que usa cruzamentos de dados o que facilita a análise das informações. Contra, não gera gráfico, os resultados são gerados em arquivo texto.

Outros softwares utilizados no desenvolvimento desse trabalho foram os de stress ou benchmarks, ferramentas que geram cargas sintéticas para testar o desempenho de sistemas.

O software Web Server Stress Tools, é de fácil configuração e gera vários gráficos, capturando recursos da máquina como uso da CPU, da memória, 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. Contra, geram cargas muito homogêneas, usam valores constante.

A ferramenta Apache Bench é uma ferramenta distribuída junto com o servidor Web Apache. Para utilizar, basta apenas utilizar o comando ab. Ferramenta de grande utilidade e simples funcionamento, roda no shell e em mais de uma plataforma. Contra, não gera gráficos, tem que interpretar os resultados.

Ferramenta W4Gen gera cargas sintéticas para modelos de servidores, possibilita gerar cargas podendo ser modificada para cada caso de estudo. Utiliza de todo os parâmetros do estudo de logs de servidores reais. Desenvolvida na linguagem Java, utiliza biblioteca PSOL, utiliza modelos matemáticos e de estatística e probabilidade para gerar uma carga de forma bem próxima das reais. Contra, só geram cargas para modelos de servidores, ou seja, servidores simulados.

Essas e outras características levantadas neste estudo são de extrema importância para que futuramente possa utilizar dessas métricas para a construção de um modelo de carga de trabalho, que é muito importante para trabalhar com parâmetros sintéticos, mas sem fugir muito das características de uma carga real, já que com modelo pode alterar parâmetros da carga para avaliar um possível resultado no servidor.

CONCLUSÃO

Com a evolução acelerada da Web, essa deixou de ser somente um meio de enviar documentos para se tornar um conjunto com milhares de tipos de objetos, usando o seu tráfego, ocasionando muitas vezes sobrecarga no sistema.

Existem inúmeras ferramentas para auxílio de diagnósticos que podem levar a projetos de sistemas melhores. Utilizando uma técnica relativamente simples de medição do tráfego em servidores Web, o estudo teve como objetivo analisar o conjunto de oito logs de servidor Web Apache reais, e aplicando para isso ferramentas que auxiliaram no processo de análise em ambiente simulado.

As vantagens que se teve em trabalhar com ambiente simulado é que não afetam o servidor em seu ambiente real, tendo que rodar ferramenta de diagnóstico poderia haver um comprometimento quanto ao desempenho do mesmo. Outro ponto é que utiliza dados retirados de servidores reais, então se entende que as requisições registradas nos logs são uma forma de preservar as principais características das cargas de trabalho. Em um intervalo de tempo menor e volume de dados maior pode-se realizar vários testes e análises e verificar como isso implicaria no desempenho do servidor.

Com um servidor rodando em seu ambiente real de trabalho poderia haver implicações em fazer teste de stress, por exemplo, esse tipo de teste pode derrubar o servidor, dependente do tipo de serviços oferecidos por esse poderia ocasionar prejuízos.

Já nas medições dos logs observou-se que com a evolução da Web, páginas do tipo dinâmicas cresceram consideravelmente, objetos poucos requisitados como os da classe dos binários e compactos, obtiveram um grande número de volume de dados transferidos, podendo haver uma sobrecarga no servidor. Também foi possível analisar a alta disponibilidade do servidor, com o código resposta da classe 2xx obtendo mais de 80%.

A partir das análises e da metodologia aplicada para a caracterização da carga de trabalho em cima dos arquivos de logs é possível estabelecer métricas e aplicá-las, para a construção de sistemas melhorados.

Para trabalhos futuros poderá usar dos resultados obtidos nesse trabalho como levantamento de métricas para a construção de um modelo de carga padrão, aplicando modelos matemáticos e estatísticos apresentados na metodologia. Assim como também a construção de ferramentas geradoras de cargas sintéticas ou de leitura de logs.

Pode-se dizer que são de extrema relevância o estudo da caracterização da carga de trabalho para uso futuro em planejamento de capacidade e desempenho de servidores Web.

REFERÊNCIAS

APACHE. Log Files. Desenvolvido por Apache Software Foundation. 2009 Disponível em: <http://httpd.apache.org/docs/2.2/logs.html>.

ARLITT, Martin F; JIN, Tai. Workload Characterization of the 1998 World Cup Web Site. Set 1999. Disponível em <http://www.hpl.hp.com/techreports/1999/HPL-1999-35R1.html>. Acesso em: 16/03/2009.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. Informação e documentação - Referências - Elaboração. Rio de Janeiro: ABNT, 2002.

CDMIL.2009. Disponível em: <http://www.cdmil.com.br/logs/access_log>. Acesso em: 03/06/2009.

COMER, Douglas E. Redes de computadores e internet: abrange transmissão de dados, ligação inter-redes e web. 2.ed Porto Alegre: Bookman, 2001.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. Ditributed System: Concepts and Design. 3. ed. Nova Jersey, EUA: Addison-Wesley, 2001.

KRISHNAMURTHY, Balachander; REXFORD, Jennifer. Redes para a web. Rio de Janeiro: Ed. Campus, 2001.

KUROSE, James F.; ROSS, Keith W. Redes de computadores e a internet: uma abordagem top-down. 3. ed São Paulo: Pearson Addison Wesley, 2006.

LOOSLEY, Chris., DOUGLAS, Frank., WILEY, John. High-Performance Cliente/Server.Rio de Janeiro: Ed. Campus, 1998.

MATTHEW, Flint Arnett. Desvendando o TCP/IP: métados de instalação, manutenção e implementação de redes TCP/IP. 4.ed Rio de Janeiro: Ed. Campus, 1997.

MENASCÉ, Daniel A., ALMEIDA, Virgílio A. F. Planejamento de Capacidade para Serviços na Web. Rio de Janeiro: Ed. Campus, 2002.

MENASCÉ, Daniel A., ALMEIDA, Virgílio A. F. Capacity Planning for Web Performance: Metrics, Models & Methods. Nova Jersey, EUA: Predice Hall, 1998.

MURHAMMER, Martin W.; GAERTNER, Jussara Licinia Souza. TCP/IP: tutorial e técnico. São Paulo: Makron Books, 2000.

SANTOS, Carla E. F. Levantamento de métricas para planejamento de capacidade de servidores Web. 2005. Projeto de Conclusão de Curso - Curso de Ciência da Computação, Universidade Federal da Bahia Instituto de Matemática, Salvador, 2005.

SILVA, Luis H. C. Caracterização de Carga de Trabalho para Teste de Modelos de Servidor Web. 2006. Titulo de Mestrado – Ciência da Computação e Matemática Computacional, Universidade de São Paulo Instituto de Ciência da Matemática e Ciência da Computação, São Carlos, 2006.

SOARES, Luiz Fernando Gomes; LEMOS, Guido; COLCHER, Sérgio. Redes de Computadores das LANs, MANs e WANs às Redes ATM. 6. ed. Rio de Janeiro: Ed. Campus, 1995.

STANFORD. 2005. Disponível em: <http://waas.stanford.edu/~wwu/logs/access_log>. Acesso em: 12/05/2009.

STARDUST. The need for QoS. White Paper. 1999. Disponível em:

<http://www.stardust.com/qos/whitepapers/need.htm>, 1999b. Acesso em: 18/09/2009.

TANENBAUM, Andrew S. Redes de computadores. 5.ed Rio de Janeiro: Ed. Campus, 1997.

TANENBAUM, Andrew S. Redes de computadores. Rio de Janeiro: Campus, 2003.

TEXEIRA, Mario. A. M. Suporte a Serviços Diferenciados em Servidores Web: modelo e algoritmos. Tese de Doutorado – Ciência da Computação e Matemática Computacional, Universidade de São Paulo Instituto de Ciência da Matemática e Ciência da Computação, São Carlos, 2004.

TRACES Available in the Internet Traffic Arquive: Disponível em: <http://ita.ee.lbl.gov/html/traces.html>. Acesso em: 12/05/2009.