AGRADECIMENTOS

Gostaria de registrar meus sinceros agradecimentos a algumas das várias pessoas que de alguma forma contribuíram para a realização deste trabalho:

Ao Prof. Dr. Carlos Augusto Paiva da Silva Martins, minha gratidão e respeito pelo profissionalismo, pragmatismo, humildade, compreensão e, principalmente, a paciência nesses anos de orientação.

À Maria Isabel Siqueira, meu carinho especial pela paciência, presteza, solidariedade e apoio incondicional em todas as atividades.

A PUC-Minas, pelo crédito, apoio e recursos investidos, sem os quais o desenvolvimento deste trabalho não teria sido possível.

Aos meus colegas do Instituto de Informática da PUC-Minas e do Programa de Pós-Graduação de Engenharia Elétrica pela colaboração, apoio, compreensão, paciência e ajuda durante todos os momentos em que necessitei.

Agradeço a Deus por sempre me acompanhar, iluminando meu caminho, dando-me paciência, força e equilíbrio.

Agradeço principalmente a minha família, em especial a minha esposa Rosamaria Menegaz Pereira da Silva e minha mãe Amélia Maria da Silva Barbosa pelo apoio e amor dedicado durante este percurso.

RESUMO

A utilização da técnica de *web-caching* vem sendo amplamente difundida com o objetivo de reduzir o impacto de alguns problemas causados pelo crescimento vertiginoso da Internet. A função principal dessa técnica é manter objetos da Internet mais próximos do usuário, conseqüentemente reduzindo a latência na resposta e o tráfego nos canais de comunicação. Uma propriedade importante dos servidores de *web-caching* é a capacidade de cooperação que resulta em aumento no desempenho em comparação a servidores isolados, porém, na maioria dos casos, a quantidade de alternativas para estabelecer esta cooperação é grande e a escolha da melhor opção pode ser complexa e árdua. Nossa pesquisa propõe um método para análise de desempenho de estruturas hierárquicas de servidores *web-caching* seguindo um modelo de experimentação em laboratório, tendo em vista a dificuldade, em determinados casos, de realizar esta análise com modelo analíticos ou através de monitoramento em ambiente real. Além disso, apresentamos a validação do método proposto aplicando-o a um estudo de caso real.

ABSTRACT

The use of the web-caching technique has been widely spread out with the objective of reducing the impact of some problems caused by the vertiginous growth of the Internet. The main function of this technique is to keep the objects of the Internet next to the user and consequently, reducing the reply latency and the traffic in the communication channels. An important property of the web-caching servers is the cooperation's capacity that results in the performance increase in comparison to isolated servers. However, in most cases, the amount of alternatives to establish this cooperation is large and the choice of the best option may be complex and arduous. Our research propose a method for the performance analysis of hierarchic structures of web-caching servers based on an experimental model, due to difficulty, in some cases, to perform such analysis with analytical models or through measurements in real environments. We also present a validation of the proposed method, applied to a real case study.

SUMÁRIO

LISTAS DE FIGURAS

LISTAS DE TABELAS

CAPÍTULO 1 - INTRODUÇÃO

1.1 Contexto

Nos últimos anos, a Internet tem evoluído de forma espantosa. Sua aceitação foi muito rápida e são diversas as aplicações desenvolvidas para este que se transformou em um gigantesco meio de acesso à informação e comunicação, interligando milhares de indivíduos e instituições com os mais diferentes objetivos [69].

A diversidade de serviços (documentos, imagens, arquivos, comércio eletrônico, vídeo, voz, etc.) e a especialização das aplicações têm promovido ainda mais o crescimento vertiginoso da Internet, que teve seu início motivado por questões estratégicas do exército norte americano. A princípio pretendia-se descentralizar o conjunto de informações em várias bases geograficamente distribuídas e distantes, evitando assim, possíveis perdas de informações [26] [42].

No segundo momento, as universidades utilizaram deste novo meio de comunicação para desenvolver novas pesquisas, divulgar resultados obtidos, promover um relacionamento mais estreito entre os grupos de pesquisas e disponibilizar informações relevantes [69].

Esta interconexão entre estes poucos centros deu origem a uma espinha dorsal (*backbone*) que foi crescendo gradativamente, possibilitando uma expansão para outros países, outros continentes e hoje conta com milhares de nós distribuídos por todo mundo, servindo para os mais diversos fins, desde provedor de informação até ponto de acesso para usuários finais.

Porém, como conseqüências deste crescimento muito rápido, surgiram alguns problemas, principalmente devido à elevada demanda gerada pelos usuários e pela especialização das aplicações, que a estrutura atual não está sendo capaz

de atender com eficiência. Entre esses problemas, podemos enumerar alta latência nas respostas para os usuários, redução na disponibilidade das informações, utilização incorreta de recursos, alta carga nos servidores, etc [26].

A elevada ocupação da Internet tem como principal efeito negativo a demora na carga de objetos requisitados por usuários, ou seja, alta latência. Isto provoca prejuízos em vários segmentos, como por exemplo, o comércio eletrônico que pode sofrer grandes perdas durante as transações realizadas em função da lentidão e falhas da Internet [42].

A redução da disponibilidade da informação é outro problema grave. Considerando que o *backbone* Internet é formado por milhares de nós, ou seja, alta capilarização, a possibilidade de queda da comunicação entre alguns *sites* (localidades) aumenta, causando isolamento de parte da rede do *backbone* global. Em outras palavras, os servidores podem ficar inacessíveis, por queda em determinado(s) circuito(s) de comunicação.

Outro problema a ser citado é o desperdício de recursos com tráfego da mesma informação em longos circuitos. Uma vez acessado um objeto, seria interessante manter uma cópia do mesmo mais próximo dos usuários para que em acessos futuros não fosse necessária uma nova visita ao servidor de origem, deixando o canal de comunicação disponível para outras requisições.

Os problemas da Internet não se restringem a custo, disponibilidade e satisfação dos usuários, mas estes motivos são mais que suficientes para justificar soluções que amenizem as deficiências observadas.

Como proposta para redução no impacto causado pela falta de recursos, alta demanda de serviços, necessidade de aumento da disponibilidade da informação e atendimento das expectativas dos usuários, surgiu à idéia da implementação de servidores *cache* de objetos *web,* ou ainda, servidores *web-*

caching. Por sua vez, esta técnica tem suas vantagens, desvantagens e problemas que serão discutidas no Capitulo 2.

1.2 Importância e Relevância de W*eb-Caching*

A Internet é uma realidade e os problemas causados por sua rápida disseminação são conhecidos e foram discutidos na seção anterior. Como já mencionamos, uma técnica proposta para amenizar estes problemas é a implantação de servidores *web-caching*. Em alguns casos também se percebe a necessidade de implantação de estruturas hierárquicas cooperativas, devido ao volume de transações ser muito grande e as estações de trabalho estarem física e logicamente muito distantes. Vale ressaltar que esta técnica não é apenas uma proposta de difícil implantação. Sua utilização é viável e a grande maioria das instituições já oferece este serviço a seus usuários. Os problemas provocados com esta técnica podem ser verificados no Capítulo 2. Estaremos utilizando, a partir desse capítulo, as palavras *colaborativa* e *cooperativa* com o mesmo significado, representando a capacidade de interação entre os servidores de *web-caching*.

A solução de w*eb-caching* hierárquico pode ser diferente para cada ambiente computacional. Esta variabilidade dificulta o projeto da estrutura e nossa pesquisa procura diminuir esta barreira.

O problema estudado é relevante por considerar um ambiente em plena ascensão, a Internet, onde a preocupação com a otimização de recursos e atendimento das expectativas dos usuários é crescente e de difícil análise.

1.3 Motivação e Justificativa

A demanda crescente pelo uso de Internet na instituição onde trabalhamos foi o primeiro fator que motivou nossas pesquisas. Era completa a falta de controle sobre o comportamento dos usuários e dos recursos da rede de computadores.

Não tínhamos conhecimento de como, quem e para que a Internet era utilizada. A insatisfação era generalizada e a utilização incorreta dos recursos predominava.

Ao consultarmos outras instituições e empresas verificamos que os problemas eram semelhantes e que em alguns casos uma solução estava sendo aplicada: servidores de w*eb-caching*.

Então nossa principal motivação foi à possibilidade de pesquisar uma solução eficiente que poderia ser utilizada em um ambiente computacional de grandes proporções com resultados e conclusões práticas que poderiam ser aproveitadas em outras situações, não ficando restritas ao contexto de nosso estudo de caso.

Os resultados (vantagens e ganhos) de nosso estudo de caso, assim como os resultados apresentados nas referências citadas durante a dissertação justificam a necessidade de estudos que promovam a evolução da técnica de w*eb-caching*.

1.4 Objetivos

Nosso objetivo geral é apresentar um método de análise de desempenho de estruturas hierárquicas de servidores w*eb-caching*, baseado na técnica de experimentação em laboratório, que oriente o processo de análise de desempenho de ambientes reais ou hipotéticos.

Outros objetivos específicos derivam da proposta inicial, entre eles: otimização na utilização de recursos; redução na latência nas respostas para os usuários; aumento na disponibilidade das informações; e controle de acesso. Algumas referências que discutem mais detalhadamente são [25] [38] [53] [66].

A otimização dos recursos tem por objetivo reduzir custos com *bandwidth* (largura de banda de comunicação de dados) mantidos pelas empresas e instituições para a Internet e na Internet como um todo, além de possibilitar um melhor dimensionamento de *hardware*.

Outro objetivo que decorre diretamente do objetivo geral é o aumento de disponibilidade dos objetos e a redução no tempo de resposta às requisições feitas pelos usuários, atendendo aos anseios e prestando, assim, um serviço de melhor qualidade.

Com a adoção desta técnica também é possível manter controle de origem e destino dos acessos e também o monitoramento das atividades dos usuários. Este monitoramento permite verificar tendência de conteúdo que poderá ser aproveitada para configuração dos servidores e conseqüente melhora no tempo de resposta às requisições.

1.5 Contribuições

A principal contribuição esperada na pesquisa é a apresentação de um método específico para análise de desempenho de estruturas hierárquicas de servidores w*eb-caching*. Este método pode servir tanto para pesquisas futuras como para aplicações práticas em ambientes computacionais. Em nosso estudo do estado da arte não identificamos nenhuma pesquisa que tivesse como resultado um método com o mesmo propósito apresentado nesta dissertação.

Esperamos também contribuir com a apresentação das experiências acumuladas durante o desenvolvimento do método e sua aplicação a um estudo de caso.

1.6 Limitações da Pesquisa

Esta pesquisa foi limitada a apresentar um método para análise de desempenho de estruturas hierárquicas, distribuídas ou híbridas de servidores w*eb-caching* seguindo a técnica de experimentação em laboratório. Para efeito de simplificação utilizaremos apenas a palavra hierárquica para representar os três tipos de arquiteturas existentes.

1.7 Estrutura da Dissertação

No Capítulo 1 apresentamos o contexto onde a pesquisa está inserida, os principais problemas encontrados, a relevância e motivação que determinaram a seleção do tema, os objetivos delineados no início dos estudos, as contribuições esperadas com o fechamento da dissertação e as limitações da pesquisa.

O Capítulo 2 apresenta o estado da arte em servidores w*eb-caching*. Na introdução são apresentadas uma definição de w*eb-caching* e algumas considerações gerais, em seguida alguns tipos de estruturas colaborativas são caracterizadas. Ainda neste capítulo são apresentados os principais conceitos e campos de pesquisa de servidores w*eb-caching*, em função de análise de desempenho. Algumas ferramentas de análise de desempenho são citadas e uma carga de trabalho é caracterizada quantitativamente a partir de várias referências. Por fim, são mencionados alguns trabalhos relacionados a essa pesquisa. Ou seja, este capítulo apresenta o contexto onde nossa pesquisa está inserida.

No Capitulo 3 o método proposto na pesquisa é apresentado e todas as etapas, relações e modelos são descritos. A partir de um fluxograma apresentado é possível identificar todos os elementos que compõem o método e qual é a relação entre cada um deles. A descrição das etapas, modelos e componentes é feita de forma minuciosa, em alguns casos exemplificando com

situações reais. Sugestões de trabalhos futuros são citadas e o capítulo termina com uma conclusão.

No Capítulo 4 aplicamos o método proposto a um estudo de caso real na rede acadêmica da PUC-Minas. As etapas mais importantes do método foram realizadas e resultados quantitativos enriquecem o capítulo. Além disso, todas as medidas adotadas para efetuar as experimentações são descritas por completo, o que permite a reprodução das simulações em outro ambiente. O objetivo geral desse capítulo é validar o método proposto, comprovando que as ações definidas no método são capazes de orientar o trabalho de análise de desempenho de servidores w*eb-caching* em estruturas hierárquicas. Algumas conclusões são apresentadas em relação às experiências acumuladas durante a aplicação do método proposto.

No Capítulo 5 apresentamos a conclusão geral da pesquisa. São discutidos os resultados considerando as principais vantagens e desvantagens do método. Os sucessos e insucessos da pesquisa são mencionados e servem como orientação para pesquisas futuras. Os objetivos da pesquisa são analisados e uma conclusão geral é apresentada. Por fim, indicamos as principais contribuições, limitações dessa pesquisa e principais trabalhos futuros.

CAPÍTULO 2 – WEB-CACHING: CONCEITOS E TRABALHOS RELACIONADOS.

2.1 Introdução

Web-caching é uma técnica que gerencia as requisições feitas por navegadores de rede e armazena temporariamente os objetos *web* recuperados com o objetivo de evitar consultas futuras aos servidores de origem desses objetos. Portanto, um servidor de *web-caching* é um sistema computacional que implementa esta técnica de gerência, normalmente dedicado e também conhecido como Proxy [25] [26] [96].

Em DAVISON [26] são apresentados os motivos que induziram o surgimento dos servidores de *web-caching*. Estes motivos decorrem dos problemas gerados pelo crescimento da Internet citados no Capítulo 1. Além disso, podemos encontrar uma definição e descrição do mecanismo básico de funcionamento no qual se baseia um aplicativo de *web-caching*. Também são apresentados os benefícios e potenciais problemas encontrados pelos pesquisadores e desenvolvedores da técnica. Este mesmo autor centraliza na referência [25] vários *links*, publicações, análises, comparações e orientações que oferecem uma ótima fonte de informações e auxilia muito nas pesquisas que abordam este assunto.

O conceito de *web-caching* deriva da solução de *cache* de dados e instruções implementada na área de arquitetura de computadores. Esta técnica reduz acessos a dispositivos mais lentos, permitindo a execução de algumas instruções mais rápida. O *cache* de dados e instruções é formado por uma memória volátil com tempo de acesso menor que outros dispositivos de memória. Seu conteúdo é, na maioria dos casos, gerenciado usando um algoritmo LRU que aumenta o número de acertos (o acerto é obtido com requisições que são atendidas sem necessitar de novos acessos a memória

relativamente mais lentas) automaticamente, deixando a execução de um processo completo mais rápido [68]. O armazenamento de objetos *web* segue a mesma filosofia, porém o conteúdo pesquisado é armazenado em memória volátil e não volátil, recebendo um tempo de vida e servindo para atender requisições durante este período de vida estabelecido. Neste caso, o tempo de resposta de uma consulta ao *web-cache* é menor do que aquele obtido ao se consultar o *site* de origem da informação desejada.

Web-caching é efetivo porque muitos recursos são requisitados freqüentemente por vários usuários, ou repetidas vezes por um usuário específico. Esta característica é conhecida como localidade de referência temporal. Uma caracterização dessa localidade de referência é discutida em [3].

Uma desvantagem desta técnica de Web-Caching é a possibilidade de aumento na latência de resposta ao usuário, em determinados casos, se compararmos a busca realizada pelo navegador do usuário direto à origem do objeto.

Uma transação em ambiente de produção com servidores *Web-caching* segue um ciclo que inicia quando um usuário requisita um objeto e termina quando o servidor entrega este conteúdo solicitado.

O ciclo completo de recuperação começa quando o navegador utilizado pelo usuário requisita um objeto ao servidor de *web-caching*. Este, por sua vez, providencia a tradução do endereço URL e verifica em sua base se o conteúdo solicitado se encontra localmente armazenado. Caso o conteúdo tenha uma cópia armazenada localmente, o servidor *web-caching* retorna este objeto ao usuário. Caso contrário, encarrega-se em recuperar o material do endereço origem e entregá-lo ao cliente, mantendo uma cópia local para possíveis acessos futuros [47] [97] [98].

No caso do objeto estar armazenado localmente, é necessário acionar o mecanismo de coerência para verificar a sua validade. Este processo será descrito posteriormente. Nos interessa no momento saber que o tempo necessário para checagem é muito menor que a recuperação do objeto inteiro, o que justifica a utilização de servidores *web-caching*.

O servidor de *web-caching* não tendo a informação armazenada localmente, providencia a recuperação do objeto *web*, solicitando-o para outro servidor de *web-caching* ou recuperando-o no servidor onde originalmente está o objeto. A possibilidade de recuperação do objeto de outro servidor *web-caching* introduz o conceito de estrutura hierárquica ou distribuída que possibilita criar colaboração entre dois ou mais servidores como tentativa de otimizar o processo de recuperação de objetos na Internet.

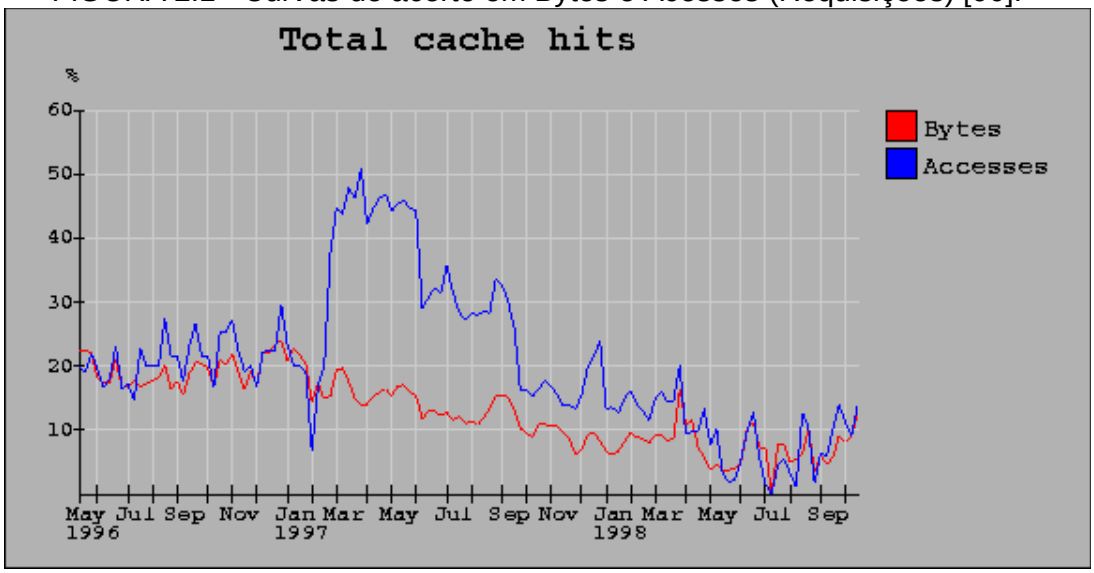
Vários parâmetros podem ser configurados com o intuito de melhorar o desempenho e a quantidade de acertos locais na *web-cache* mantida pelo servidor, mas a configuração destes parâmetros dependerá da implementação utilizada, estrutura do sistema, modelo de uso da *web* e também dos objetivos delineados pela organização. Alguns índices de desempenho são discutidos adiante.

Uma lista mais completa de produtos que têm como uma ou principal função realizar armazenamento temporário de objetos *web* pode ser encontrada em [25], porém podemos citar alguns como os mais conhecidos: CacheFlow, Cisco Cache Engine, IBM Websphere Edge Server, Inktomi Traffic Server, Microsoft Proxy [58], NetCache, NetFilter, Netscape Proxy Server e Squid [97].

Um *software* de *web-cache* muito utilizado na Internet é o Squid [97]. A sua aceitação se deve a vários motivos, podendo-se citar: o fato de ser gratuito, o que viabiliza sua adoção; exigir pouco em relação a *hardware*; e ser mais flexível que um dos seus principais concorrentes o Microsoft Proxy Server [58].

O gráfico apresentado na FIG. 2.1 mostra resultados de acertos em número de requisições e volume de dados, entre maio de 1996 e Agosto de 1998, em um servidor de *web-caching* mantido na UNINETT, um ponto de presença representativo do *backbone* mundial e que serviu para os estudos desenvolvidos em 1998 [90]. Neste gráfico podemos observar a percentagem de requisições que produziram acerto na base de dados mantidos localmente e, conseqüentemente, evitaram solicitação direta aos servidores onde estavam armazenados os objetos solicitados e também reduziram consideravelmente a latência para os usuários que dispararam as consultas.

FIGURA 2.1 - Curvas de acerto em Bytes e Acessos (Requisições) [90].



Podemos considerar outras estatísticas que mostram a eficiência da técnica de w*eb-caching*, por exemplo, a tabela de 28 de setembro de 2001 - TAB. 2.1, obtida das páginas do *site* mantido pelos administradores do ponto de presença Internet da Rede Minas. Nesta tabela podemos observar os nomes dos servidores monitorados e as taxas de acerto nas consultas (% *Hit* Objeto), ou seja, percentagem de solicitações atendidas localmente pelo servidor, dispensando consulta na Internet.

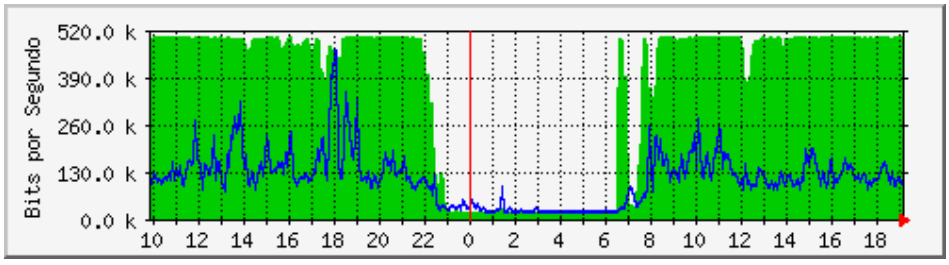
TABELA 2.1 - Estatísticas de acesso no servidor de web-c*aching* da Rede Minas, extraída no dia 28/09/2001 [73].

Nome do Servidor	% HIT Objeto
Cache.redeminas.br	22,8
Cache1.redeminas.br	13,3
Cache2.redeminas.br	28,6

Outro argumento forte a se considerar é a tendência de acréscimo em consultas repetidas ao mesmo *site* devido à atividade mantida na organização em questão. Por exemplo, em uma universidade o comportamento e as necessidades coletivas dos usuários tendem a acumular um maior número de consultas a endereços com informações acadêmicas. Esta característica normalmente eleva em cerca de 10 % a taxa de acerto nos acessos [38].

Em relação à escassez de recursos, podemos apresentar o gráfico de monitoramento do canal de comunicação para Internet da rede acadêmica da PUC-Minas, que em um período de 01 (um) ano sofreu três atualizações de largura de banda de comunicação e, mesmo assim, não tem sido suficiente para atender a demanda cada vez maior por serviços *web*. A FIG. 2.2 apresenta um gráfico de estatísticas diárias, com consultas feitas de 5 em 5 minutos ao roteador do *link* (canal de comunicação) de 512 Kbps mantido entre a PUC-Minas e a Embratel [9]. A área preenchida indica o consumo de banda no sentido Internet para Rede PUC; e o traço mais escuro indica tráfego no sentido rede PUC para Internet. A abscissa está em função do tempo e a ordenada em função da largura de banda consumida em *bits* por segundo.

FIGURA 2.2 - Curvas de ocupação do canal de comunicação para Internet da PUC-Minas.



Discutimos no capitulo anterior os motivos que induziram o desenvolvimento de servidores de *web-caching*. Nos parágrafos anteriores apresentamos a definição de *web-caching* e alguns *softwares* que assumem esta função, suas principais características e alguns exemplos quantitativos que justificam a utilização desta técnica. Porém ainda não mencionamos os problemas gerados por esta solução. Entre estes problemas podemos enumerar: determinação da estrutura mais adequada para cada contexto; identificação de índices de desempenho mais adequados para análise em cada caso; solução de busca antecipada de objetos; técnicas de roteamento; políticas de substituição de conteúdo do *cache*; e mecanismos de coerência. Estes aspectos motivaram e ainda motivam muitos trabalhos científicos como apresentaremos durante o restante deste capítulo.

2.2 Web-Caching Cooperativo

Uma das propriedades dos servidores de *web-caching*, como mencionado anteriormente, é a capacidade de colaboração ou cooperação que pode ser estabelecida entre dois ou mais servidores. Esta capacidade de cooperação permite projetar arquiteturas mais elaboradas com o objetivo de aumentar a eficiência dessa técnica de *cache*. Esta colaboração pode ser implementada com os conceitos hierárquicos, distribuídos, ou de maneira mista. Esta característica de cooperação é muito importante para nossa pesquisa, tendo em vista que nossa proposta é oferecer um método de análise de desempenho que pode ser usado na determinação da melhor estrutura colaborativa para uma entidade ou um grupo de entidades.

O estabelecimento de cooperação entre servidores de *web-caching* tem algumas vantagens e desvantagens. Podemos enumerar as seguintes vantagens: balanceamento de carga (cada servidor armazena um conjunto de objetos, facilitando a distribuição das requisições e conseqüentemente a carga nos servidores); redução no tempo de resposta ao usuário quando o objeto é localizado dentro da estrutura colaborativa; elevação da taxa de acerto;

redução de consultas nos servidores de origem. Uma desvantagem nessa colaboração é que em determinadas consultas dentro da estrutura o tempo de resposta ao usuário pode ser maior do que nos casos de uma busca direta à origem, isto devido ao atraso acumulado nas consultas entre os servidores *web-caching*.

2.2.1 Estrutura Hierárquica

Web-caching hierárquico foi inicialmente proposto no projeto Harvest em 1996 [82]. Pela definição estabelecida durante o projeto, o sistema de *web-caching* se dispunha em estrutura de árvore onde cada nodo aponta para um servidor pai no nível superior da árvore até se atingir o servidor *root* - servidor mais no topo da estrutura e que tem a função de recuperar os objetos direto dos servidores de origem. Quando um objeto é localizado, seja em um dos servidores da árvore ou no servidor origem, este percorre todo o caminho inverso até o usuário, passando pelos servidores por onde a requisição transitou. Uma cópia do objeto é mantida em cada um dos servidores intermediários. As desvantagens [9] [37] [48] [51] dessa técnica são: o atraso adicional ocasionado em cada mudança de nível da árvore; redundância dos objetos (problemas de espaço ocupado e consistência); os servidores no topo da hierarquia são pontos de contenção; longo tempo de atraso nas consultas.

2.2.2 Estrutura Distribuída

Em conjuntos de servidores com arquitetura *web-caching* distribuída não são definidos servidores em níveis intermediários. Basicamente todos os servidores encontram-se em um mesmo nível e colaboram entre si, quando um erro de acesso é verificado [48] [51] [70] [88]. Normalmente, esta técnica é adotada em redes institucionais e, pelo fato de não existir níveis intermediários, estes servidores necessitam de outro mecanismo de compartilhamento dos objetos estocados. Alguns desses mecanismos são: Internet Cache Protocol (ICP) [88] [94] [95]; Sumário [51]; Hash [76].

2.2.3 Estrutura Híbrida

No esquema híbrido, os servidores de *web-caching* cooperam entre si no mesmo nível usando *web-caching* distribuído e entre níveis usando *web-caching* hierárquico. Alguns servidores que utilizam ICP são um típico exemplo. Uma ferramenta que se enquadra neste contexto é o Squid. Nesta ferramenta os servidores utilizam o conceito de servidores pais e vizinhos ou irmãos.

Outro exemplo de esquema híbrido é a solução discutida em [23] que utiliza o Caching Neighborhood Protocol (CNP) que, em suma, é uma estrutura de *web-caching* que monta a tabela de servidores cooperados dinamicamente, de acordo com a disponibilidade dos servidores e parâmetros de desempenho.

2.3 Principais Características Avaliadas em *Web-caching*

As principais características avaliadas em *web-caching* são divididas em alguns segmentos como pode ser observado nas referências [48] [51], sendo elas: métricas de desempenho, técnicas de busca antecipada, roteamento, políticas de substituição e mecanismos de coerência. A seguir, cada uma destas características é apresentada.

2.3.1 Métricas de Desempenho

As métricas de desempenho em sistemas de *web-caching* são fundamentais para permitir o estabelecimento de modelos comparativos entre diferentes soluções. Estes índices podem ser divididos em dois grupos: um focado somente em servidores isolados e outro para arquiteturas compostas de pelo menos dois servidores. Exemplos de métricas normalmente consideradas [25] [48] [51] [55] são:

- Carga de tráfego: inclui distribuição da quantidade de carga transferida, distribuição de tamanhos dos objetos transferidos e intensidade de tráfego no *web-caching*;

- Análise de acertos: percentagem e classificação em quantidade ou em bytes, resultante de consultas locais ou na arquitetura;
- Latências: tempo de conexão do cliente e tempo de resposta do *web-caching* é o tópico principal;
- Utilização de subsistema de armazenagem em disco;
- Utilização de memória e CPU;
- Análise de carga de trabalho;
- Utilização da rede;
- Índices de saída. Inclui a avaliação da concorrência de conexões de saída, tempo de conexão do proxy ao servidor origem e tempos de resposta;
- Tráfego extra gerado na rede pelo *proxy*, principalmente para comunicação entre servidores;
- Tráfego útil gerado pelo *proxy*;
- Efeito da carga no *proxy*;

2.3.2 Busca Antecipada

Busca antecipada é uma técnica que permite aos servidores *web-caching* recuperarem um objeto da origem, antes mesmo desse objeto ser solicitado pelo usuário, utilizando algum algoritmo específico para esta função.

Todos os servidores de *web-caching* podem ser enquadrados dentro de duas categorias de busca antecipada: *caches* passivos (efetuam busca quanto um agente externo solicita) e ativos (acionam por decisão própria um algoritmo de busca antecipada). Uma boa técnica de busca antecipada de objeto aplicada a um servidor de *web-caching* pode representar um aumento no desempenho do *proxy*, reduzindo o tempo de latência aos clientes de 10% a 30% [51].

Os algoritmos de busca antecipada, em sua maioria, consideram os seguintes parâmetros para decidir qual é o próximo objeto a ser recuperado: modelos de busca antecipada, distribuição de popularidade, tamanho dos objetos, tempo

de vida dos objetos, recursos disponíveis para busca antecipada e ponto inicial especificado. Algumas estratégias e algoritmos mais comumente usados para tentar aumentar o desempenho dos servidores *web-caching* são: “Threshold algorithm”; predicado local de busca antecipada [92]; busca antecipada de *hyperlinks* [29]; “Top-10” [52]; técnica similar à utilizada em sistemas computacionais e de banco de dados [45]; predicado futuro [32].

Estas técnicas consideram um servidor *web-caching* isolado. Para se aplicar às estruturas hierárquicas, os algoritmos crescem em complexidade de implementação. Outros problemas são a necessidade muito maior de área para armazenamento de objetos e a elevação de consumo de banda, tendo em vista que recursos estarão sendo ocupados para recuperar objetos que podem não ser solicitados.

2.3.3 Roteamento

Os sistemas de *web-caching* compostos por vários servidores em uma arquitetura distribuída ou hierárquica aumentam a disponibilidade ou servem para aumentar a localidade física atendida. O principal desafio nestas estruturas é descobrir rapidamente a localização de um objeto desejado dentro do esquema. A abordagem mais comum em uso é subir os níveis mais altos da estrutura em árvore consultando cada um dos servidores desse nível com maior demanda de popularidade. Existem dois tipos básicos de abordagem [51] [78]: uma baseada em uma tabela de roteamento; e outra baseada em funções *hash*.

A utilização de ICP para consultar os servidores que fazem parte da tabela de rotas é uma técnica largamente usada, mas que aumenta consideravelmente o tráfego na rede [88]. Para minimizar este efeito sugere-se configurar o menor número de servidores de um mesmo nível, principalmente quando este servidor não se encontra na mesma rede física. E, logicamente, procurar um ajuste dos

parâmetros de *cache* para aumentar o acerto local e impedir que mensagens ICP sejam propagadas na rede.

Outra técnica utilizada se baseia na manutenção de sumários do conteúdo de cada um dos servidores que participam da arquitetura. Este sumário é mantido em cada um dos servidores e é consultado em todo caso de falha ao se localizar um objeto solicitado. Esta técnica é conhecida como *web-caching* adaptativo [28].

Os métodos de roteamento ICP e HASH padrão são comparados em [85] a um novo método proposto por ASAKA [7]. O método ICP e HASH padrão exigem mais de uma consulta a servidores *web-caching* em função de uma requisição disparada, sendo que com o método proposto apenas uma consulta é necessária.

2.3.4 Políticas de Substituição

Este é provavelmente o tópico mais estudado no que diz respeito à tentativa de melhoria de desempenho nos servidores *web-caching* independentes ou em estruturas colaborativas mais complexas [48] [51] [61].

A principal diferença entre *cache* e *web-cache* é a alta variabilidade de tamanho dos objetos no *web-cache* [1] [62] [63]. Em função dessa variabilidade em MURTA [63] é proposto um modelo analítico que utiliza duas métricas para avaliar políticas de substituição em *web-caching*: taxa acerto em requisição e taxa de acerto em *bytes*. Baseado em uma relação analítica estabelecida é introduzido um conceito de mapa de performance para uma dada carga de trabalho. Estes mapas servem como importante ferramenta para desenvolvimento e comparação de diferentes políticas de substituição.

Normalmente as pesquisas de substituição consideram um único servidor de *web-caching*, logo arquiteturas hierárquicas podem não apresentar o mesmo

desempenho ótimo com a mesma técnica aplicada para um *cache* simples [19]. Em MENAUD [56] existe uma preocupação maior em apresentar uma política de substituição que considera estruturas hierárquicas e não apenas um servidor web-caching isolado.

Os estudos neste tópico avaliam as políticas de substituição dos objetos armazenados nos servidores de *web-caching* com o objetivo de aumentar o desempenho do servidor, incrementando o acerto na base de objetos locais ou dentro da arquitetura completa. Políticas conhecidas e/ou propostas:

- LRU (menos recentemente usado), este é o mais amplamente utilizado e importante algoritmo de substituição desenvolvido para *caching* em memória e disco;
- "GreedyDual-Size" [16] combina tamanho do objeto, custo para recuperação do objeto e tempo de última referência, porém não considera popularidade de acesso ao objeto;
- LRV - último valor relativo, utiliza vários parâmetros para determinar o valor relativo [72];
- FBR ou LFU - substituição baseada em freqüência [74];
- LPPB-R (Least Popularity Per Byte Replacement) [43];
- PPBL (Parallel Pull-Based LRU) [18];
- LRU-SP [21];
- LRU-K and 2Q [67];
- LRU Segmentado [40];
- LRU generalizado [1];
- LRU HOT e LRU QoS [56];
- LRU-Min [56];
- LAT – "Latency Access Time" [56];
- HYB [56];
- Baseado no site [98].

Em BUSARI [15] são avaliadas as influências da heterogeneidade das políticas de substituição, particionamento das áreas de disco e particionamento

compartilhado entre servidores de níveis diferentes em uma estrutura de *web-caching* hierárquica de dois níveis.

2.3.5 Mecanismo de Coerência

Os servidores de *web-caching* precisam garantir que o objeto que está em sua base de dados local seja coerente com o objeto existente no servidor de origem deste objeto. Para tanto são utilizados diferentes mecanismos de coerência a fim de realizar esta verificação [27] [36] [48] [51].

Portanto, existe esta necessidade do servidor *web-caching* atualizar ou checar os seus objetos armazenados localmente para atender as requisições dos clientes com objetos que representem a realidade, ou seja, que estejam iguais ao objeto hospedado no site de origem. Neste aspecto o problema de coerência do *cache* em um sistema de *web-caching* é similar ao problema de *cache* em sistemas de arquivos distribuídos. Atualmente, os mecanismos de coerência podem ser divididos em três tipos: iniciado pelo cliente, iniciado pelo servidor e iniciação híbrida.

Para iniciação cliente, existem vários tipos de mecanismos para manter a coerência do *cache*, são eles:

- PER (Poll Each Read) – a cada leitura o cliente questiona ao servidor origem do objeto se este é válido;
- Algoritmo de Checagem Periódica – baseado no PER, porém assume um tempo mínimo de vida do objeto antes de consultar o servidor origem;
- Validação do *cache* sobreposto [47];
- Tempo de Vida (TTL)– mantêm consistência do *cache* utilizando o atributo tempo de vida do objeto;
- TTL adaptativo [50] – ajusta o TTL do objeto baseado na observação da vida do objeto.

Mecanismos de coerência iniciados pelo servidor:

- "Callback" [37] [65] – Servidores mantêm informações de quais clientes armazenaram quais objetos, quando os objetos são alterados os servidores informam aos clientes;
- PSI [46] – são designados volumes a grupos de servidores, cada volume recebe um identificados e uma versão corrente, quando os objetos são atualizados estes volumes recebem notificações.

Mecanismo de iniciação híbrida integra iniciação cliente e servidor. Alguns algoritmos que se enquadram nesta categoria são abordagem de aluguel [34], aluguel de volumes [99] e aluguel adaptativo [30].

2.4 Carga de Trabalho

Carga de Trabalho ou ainda *workload* no contexto de *web-caching* é a designação atribuída a um conjunto de requisições de diferentes objetos que será utilizada para promover carga em um ambiente que, em nosso caso, pode ser um servidor de *web-caching* ou uma estrutura hierárquica de *web-caching*.

Em ARLITT [5] é apresentado um estudo detalhado de caracterização de carga de trabalho (*workload*) do site World Cup Web 1998. Acessos ao site foram coletados durante 3 meses, acumulando 1.35 bilhões de requisições. Examinando os resultados coletados e comparando com caracterizações de outros estudos foi possível avaliar como a carga nos servidores *web* tem evoluído. Foi detectado que com a elevação na estrutura hierárquica de *web-caching* a carga nos servidores *web* foram alteradas.

Uma caracterização quantitativa razoável pode ser estabelecida em função de tamanho e distribuição de objetos [3] [6] [13] [24] [35], carga nos servidores [84], origem das requisições [6], como apresentado por AGGARWAL [1] e reproduzida na TAB. 2.2.

TABELA 2.2 - Caracterização sumarizada de carga de servidores web [1].

Característica	Referência	Detalhe
Carga do servidor está crescendo exponencialmente	Seltzer [84]	Requisições e objetos armazenados mostram crescimento exponencial
A maioria das requisições é originada de clientes remotos	Arlitt [6]	Clientes Remotos: ≥ 70% de requisições e ≥ 60% de bytes.
Distribuição da latência é alta.	Mogul [59]	Aparece nos log normais
A maioria das transferências é pequena	Arlitt [6] Almeida [3]	Tamanho médio < 21 KB Imagem 13 KB, Texto 4 KB, Áudio 179 KB e Vídeo 2300 KB.
Distribuição do tamanho de objetos segue "Pareto"	Arlitt [6] Crovella [24]	0,40 < α < 0,63 para objetos transferidos α = 1,06 para objetos transferidos
A maioria das transferências são imagens ou HTML	Arlitt [6] Gwertzman [35] Almeida [3]	Imagem 36-78% HTML 20-50 % e Dinâmico 0-7% Imagens 65% HTML 22% Dinâmico 9% Imagens 75% HTML 19% Dinâmico 0% Áudio 5% Vídeo 0,4%
Popularidade de objetos não é uniforme	Arlitt [6] Bestravos [13]	10% dos objetos enviados satisfazem 90% das transferências 5% dos bytes enviados satisfazem 85% do tráfego em bytes
A maioria das transferências é duplicada	Arlitt [6]	> 97% enviadas mais de uma vez
Objetos têm longo tempo de vida	Bestravos [13]	JPEG 100 dias GIF 85 dias HTML 50 dias
Popularidade varia pela região	Seltzer [84]	Clientes tendem a acessar servidores geograficamente relacionados
Popularidade pode mudar rapidamente	Seltzer [84]	Popularidade de objetos e servidores pode mudar muito rápido, produzindo "flash crowds"

Algumas armadilhas comuns ao se elaborar uma carga de trabalho são discutidas em BANGA [8], entre elas podemos citar: inabilidade em gerar carga; alta variabilidade de atraso da Internet (WAN); capacidade de recursos dos clientes interferindo nas medições do servidor, ou seja, estações clientes transformando-se em gargalo na simulação; carga excessiva aplicada aos servidores não condizente com carga real. No mesmo trabalho é apresentado

um método escalar para geração de requisições *web* que em seguida é aplicado a um estudo de caso.

Em GWERTZMAN [36] é apresentada uma caracterização de objetos web baseado em seus tipos e classes. Nove servidores foram analisados e três tipos distintos de classes definidas: educacional, comercial e de noticias. Os resultados indicaram significativa diferença entre objetos estáticos e dinâmicos. A eficiência dos *web-caching* pode ser aumentada trabalhando-se os TTL dos objetos de acordo com sua classificação. O artigo também provê um guia para projeto e desenvolvimento de técnicas de *web-caching* e busca antecipada que podem explorar as características dos objetos *web* em diferentes ambientes.

Em LINDEMANN [49] é apresentado um relatório técnico dividido em duas partes. Na primeira parte são apresentados um estudo de diferentes implementações de *web-caching* e características futuras de *workload*. Esta caracterização é apresentada de forma quantitativa com objetos do tipo HTML, imagens e multimídia. Também são discutidas técnicas de substituição de objetos em *cache*. Na segunda parte é apresentado um estudo de desempenho de quatro protocolos cooperativos de web-caching e a partir desses estudos algumas conclusões são obtidas com objetivo de orientar os provedores de serviços Internet, clientes *web* e provedores de serviços de aplicação.

2.5 Benchmarks

Benchmarks para servidores *web-caching* são ferramentas (softwares) que simulam a existência de processos clientes e servidores que auxiliam na tarefa de simulação de carga em servidores *web-caching*. Tais ferramentas também possibilitam a extração de valores quantitativos de índices de desempenho que permitem a análise de esquemas de servidores *proxy*. Algumas ferramentas que realizam esta tarefa são: WebPolygraph [79]; Surge [12]; Webpacity [93]; http Blaster [91]; WebJamma [39]; Winsconsin Proxy Benchmark [2]; Proxycizer [33]; httperf [60]; Medusa Proxy [44].

2.6 Trabalhos Relacionados

Em RODRIGUEZ [75] podemos encontrar um modelo matemático proposto para analisar alguns importantes parâmetros de desempenho como: latência percebida pelos clientes, uso de banda de comunicação, carga no *cache* e uso de espaço em disco. Este modelo pode ser aplicado em todos os três esquemas de arquiteturas cooperativas apresentadas. De acordo com os resultados, no aspecto de latência, sistemas hierárquicos de *web-caching* possuem menor tempo de conexão enquanto *web-caching* distribuído tem um menor tempo médio de transmissão. Em uso de banda de comunicação, *web-caching* hierárquico tem menor uso de banda, enquanto que *caching* distribuído irá distribuir melhor o tráfego do que irá usar banda de comunicação. Para acesso ao disco, o *web-caching* distribuído terá um custo muito menor do que o *web-caching* hierárquico. Também foi encontrado que sistemas de *caching* distribuído podem compartilhar muito bem o total da carga do sistema e não gerar pontos críticos com alta carga. Em relação a *caching* hierárquico e distribuído, os autores também fizeram uma comparação de desempenho do esquema híbrido, onde *caches* cooperados em cada nível da hierarquia usavam estrutura distribuída. O artigo indica que a eficiência da latência varia dependendo do número de *caches* que cooperam em todo nível da rede e é determinado o número ótimo de *caches* que devem cooperar em todos os níveis para minimizar a latência experimentada pelos clientes.

Em BRANDÃO [14] é apresentado um simulador de *web-caching* capaz de executar simulações de arquiteturas distribuídas e cooperativas. O simulador permite medir o impacto das configurações de parâmetros no desempenho do *proxy*, permitindo fazer o projeto da melhor arquitetura cooperativa de *web-caching*. Além disso, é apresentado um estudo de caso da Rede Nacional de Pesquisa (RNP). Neste estudo de caso são simulados os resultados para variações aplicadas no método de substituição de objetos, áreas em disco reservadas para armazenamento de objetos e na disposição dos servidores de *web-caching*.

O artigo de CHE [19] objetiva apresentar os princípios fundamentais para projeto de *web-caching* hierárquico. Uma técnica de modelagem analítica é desenvolvida para caracterizar um sistema *web-caching* de dois níveis cooperativos que utiliza LRU para substituição. Com esta modelagem foi possível afirmar que objetos com freqüência de acesso (taxa em tempo de acesso do objeto) abaixo da freqüência de corte (taxa em tempo de permanência do objeto no cachê) têm boas chances de causar um acerto na consulta no conteúdo do servidor *proxy*. Com este princípio apresenta-se um algoritmo de projeto de *caching*, e em seguida é definida uma estrutura que segue este algoritmo. As simulações estudadas mostram que o esquema cooperativo proposto resulta em 50% de acréscimo no aproveitamento de recursos do *cache* se comparados com os tradicionais sistemas de *caching* não cooperativos.

Uma abordagem diferente em relação à caracterização do tipo de *web-caching* é discutida em BARISH [10]. Esta discussão leva em consideração a localização do *cache* na Internet: próximo ao cliente, próximo à origem ou estrategicamente localizado. Os tipos definidos são: Proxy Caching, Proxy Caching reverso; Caching transparente; Web-caching adaptativo; Web-caching ativo; *push caching*.

Outra abordagem caracteriza os servidores de *web-caching* em função dos métodos de descoberta, disseminação e entrega [28]. São efetuados testes comparativos entre categorias de taxonomia e características Web, onde se usa a taxonomia para classificar os atuais projetos de *proxy*. Depois é apresentado um protocolo cooperativo particular de *web-caching.* Nesse protocolo cada servidor *proxy* efetua uma pesquisa em um diretório de metadados a procura do objeto desejado. Este diretório de metadados é propagado entre os servidores permitindo uma consulta direcionada apenas ao servidor que mantém o objeto desejado. Um modelo de carga sintético analítico derivado empiricamente é usado para simular o protocolo e os resultados

indicam uma substancial redução na carga dos servidores e quebras de conexão quando comparados aos modelos atuais.

Em CHIANG [23] é apresentada uma modelagem que avalia desempenho de estruturas hierárquicas de *web-caching* em relação ao parâmetro: tempo de resposta. Os modelos avaliam estruturas com hierarquia simples, *cache* sumário, ICP e CNP. São apresentados alguns fluxogramas interessantes que representam os processos dos esquemas de *web-caching*.

Em TEWARI [88] são descritos o projeto e a implementação de uma arquitetura integrada para os sistemas *web-cache* que escalam de centenas aos milhares de *caches* e de milhares aos milhões de usuários. Não só verificando a taxa de acerto, a pesquisa avaliou transações fim a fim considerando tempos de acertos e de erros. Foram verificados vários *web-caching* e cargas de trabalho, e dessa análise foram definidos três princípios para escalar grandes sistemas de *caches* distribuídos: minimizar o número de saltos para localizar e acessar um objeto tanto no acerto como no erro; compartilhar objetos entre muitos usuários e escalar para vários *caches*; armazenar objetos perto dos clientes. Baseado nestes princípios, nem sempre observados, foram projetadas, simuladas e implementadas duas estratégias de organização de *caching* que apresentaram resultados melhores quando comparados com uma estrutura tradicional de *web-caching* hierárquico.

A dissertação de mestrado de FONSECA [31] e o artigo [53] apresentam uma nova abordagem para analisar desempenho de servidores *cache* com base em duas métricas: taxa de acerto no nível de hierarquias e eficiência computacional. Esta abordagem permite aos usuários quantificar compromissos entre configurações, facilitando a calibração de hierarquias de *cache*. A utilização desta abordagem é ilustrada através da análise de possíveis configurações para uma hierarquia de servidores *cache*. Para medir a eficiência computacional, os autores procederam com a instrumentação do código fonte do Squid. A instrumentação do código possibilitou a extração dos

coeficientes de desempenho utilizada nas comparações estabelecida para o estudo de caso do POP-MG.

2.7 Conclusões

O estado da arte de *web-caching* é muito amplo e existem muitos tópicos relacionados, o que dificulta uma apresentação mais detalhada de todos os tópicos e características. Além disso, devemos considerar as limitações de tempo, espaço e a consideração dos objetivos e metas dessa dissertação.

O que procuramos apresentar é uma síntese dos principais tópicos e chamar a atenção para os itens mais relevantes para nossa dissertação. Por exemplo, os índices de desempenho e os tipos de estruturas cooperativas.

CAPÍTULO 3 - PROPOSTA DE MÉTODO PARA ANÁLISE DE DESEMPENHO DE ESTRUTURAS HIERÁRQUICAS DE SERVIDORES WEB-CACHING

3.1 Introdução

Podemos identificar três técnicas diferentes para realizar análise de desempenho de servidores hierárquicos de w*eb-caching*: através de um modelo analítico, através de medição ou monitoramento em ambiente real e por meio de experimentação em laboratório (usando simulação) [81].

O modelo analítico permite definir, através de funções, a relação entre os parâmetros da estrutura hierárquica de servidores de w*eb-caching* e os critérios de desempenho escolhidos, em termos de equações que podem ser resolvidas analiticamente [86]. À medida que a complexidade do sistema aumenta, a dificuldade para a formulação do modelo também aumenta e, em certos casos fica muito difícil, ou até impossível, chegar ao modelo analítico útil ou preciso. Esta técnica, aplicada em sistemas pouco complexos, tem um custo de desenvolvimento baixo e pode ser a melhor alternativa.

A medição ou monitoramento em ambiente real tem a vantagem de trabalhar com carga real de trabalho que representa de forma exata as características dos usuários do ambiente e do próprio sistema como um todo. A carga de trabalho neste tipo de avaliação também pode ser gerada de forma sintética, mantendo a infra-estrutura do ambiente real, com a vantagem de reproduzir situações que poderiam demorar a surgir ou mesmo com o intuito de reduzir o tempo de análise.

No entanto, a abordagem em ambiente real, tanto com carga sintética como real, é a menos indicada para análise de desempenho de servidores de w*eb-caching*, isto por que: a experimentação deve ser mais longa; os resultados de

uma estrutura podem estar viciados, devido a alguma mudança de comportamento dos usuários em determinado período; maior dificuldade em manipular a estrutura hierárquica de w*eb-caching* no ambiente de produção [81].

A experimentação em laboratório é a abordagem mais adequada para as situações em que a utilização de um modelo analítico se mostra inviável, e por ser bem mais flexível que a medição ou monitoramento realizado em ambiente real. Nesta abordagem, o tempo de experimentação pode ser consideravelmente reduzido, o espaço da pesquisa e análise podem ser melhor planejados, e os parâmetros de carga podem ser manipulados, permitindo simular características que poderiam demorar a surgir em ambientes reais com carga de trabalho real e prevendo variações de comportamento dos usuários e do próprio sistema [55] [71] [81].

Em ambiente real, assim como na experimentação em laboratório, podemos avaliar os parâmetros obtidos a partir da instrumentação do sistema que realiza a função de servidor de w*eb-caching* ou simplesmente monitorando os arquivos de eventos gerados pela implementação de *web-caching*. A instrumentação pode ser muito útil e até mesmo necessária para modelar ou analisar o comportamento de determinadas variáveis que, por padrão, não estariam a disposição para análise [31].

Procuramos propor, neste trabalho, um método que se aplique em diferentes situações que envolvam a análise de desempenho de estruturas hierárquicas de servidores w*eb-caching* realizadas usando a técnica de experimentação em laboratório. Este método é mais específico do que a proposta de análise de desempenho de estruturas Cliente/Servidor de MENASCÉ & ALMEIDA [55], mas segue o mesmo paradigma Cliente/Servidor. Podemos caracterizar nosso método como sendo mais específico por tratar elementos comuns e importantes a servidores de w*eb-caching* que não são encontrados em todas

as estruturas Cliente/Servidor, além de definirmos etapas direcionadas ao ambiente objeto de nossos estudos.

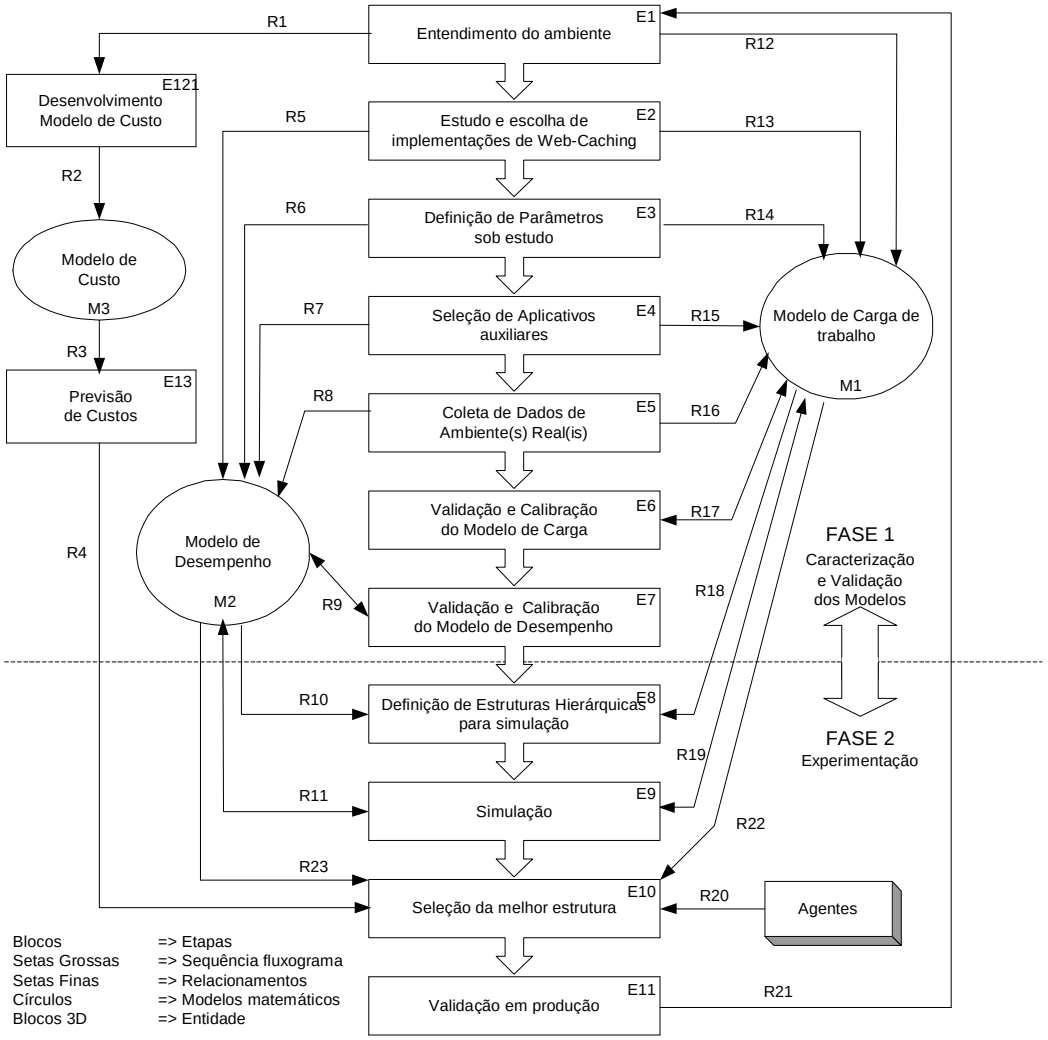
A complexidade da tarefa de análise de desempenho ou planejamento de capacidade de um sistema de computação são assuntos que motivaram o surgimento de diversos trabalhos. Algumas referências são mencionadas no Capítulo 2. Em nosso caso, a motivação principal (apresentada no Capítulo 1) foi atender uma necessidade apresentada por diversos segmentos com o mesmo propósito: como proceder a análise de desempenho de estruturas hierárquicas de servidores w*eb-caching*.

O método proposto é fundamentado em etapas/ações (blocos), modelos matemáticos (círculos) e relacionamentos (setas finas) (FIG. 3.1) e verificamos uma possibilidade de automatização da execução do método, facilitando a análise de desempenho em diversos casos.

Assim como descrito em MENASCÉ & ALMEIDA [55], os dois métodos se baseiam no uso de três modelos: um modelo de carga de trabalho (M1), um modelo de desempenho (M2) e um modelo de custo (M3). Os métodos apresentam diferenças nas etapas/ações que antecedem a formulação dos modelos, sendo que no método proposto as etapas são mais específicas considerando-se o ambiente em estudo. Uma característica distinta de nossa pesquisa é a fase de experimentação (FIG. 3.1) que é descrita de forma minuciosa e direcionada ao contexto apresentado.

Outra diferença a ser citada é em relação à preocupação em considerar, no método proposto, demanda e expectativa gerada pelos grupos envolvidos em todo o ambiente em estudo, estabelecendo um nível de compromisso entre eles. Esta característica é representada pelo bloco denominado “Agentes”.

FIGURA 3.1 - Método para Análise de Desempenho de Estruturas Hierárquicas de Servidores *web-caching*.



A FIG 3.1 apresenta o método e todos os componentes (etapas, modelos, relacionamentos) que o formam. Esta identificação serve para associar passagens do texto à figura, facilitando o acompanhamento e entendimento. A simbologia utilizada denomina as etapas com a letra 'E', os modelos com a letra 'M' e os relacionamentos (interação entre componentes) com a letra 'R', possibilitando composições mais significativas. Por exemplo, E6.R17.M1 que representa a seta de relacionamento entre a etapa de validação e calibração do modelo de carga (etapa 06) e o modelo de carga de trabalho (modelo 01), no sentido etapa para modelo, ou ainda, M1.R17.E6 similar ao exemplo anterior invertendo o sentido do relacionamento, agora do modelo para etapa.

Durante a descrição das etapas 06 e 09 são apresentados fluxogramas mais detalhados em que a simbologia utilizada recebe novos elementos: sub-etapas são apresentadas com a letra ‘e’ (retângulos); relacionamentos internos com a letra ‘r’ (setas) e etapas de teste com a letra ‘m’ (losângulos). Além de auxiliar o leitor esta simbologia auxiliará uma possível implementação automática em *software* do método.

O método pode ser dividido em duas fases, a primeira onde são feitas a caracterização e validação dos modelos de carga de trabalho, desempenho e custos, na segunda fase são realizadas as experimentações. Estas fases são apresentadas na FIG.3.1 como FASE 1 e FASE 2 e uma linha pontilhada identifica a separação entre as duas.

3.2 Caracterização e Validação dos Modelos

Nesta fase os modelos de carga de trabalho, desempenho e de custo são definidos. Descrevemos as etapas/procedimentos que precedem a formulação desses modelos, discutindo a importância de cada atividade e a sua contribuição no transcorrer da modelagem. Depois apresentamos as funções, os parâmetros e outros componentes que formam os modelos.

As definições e considerações para caracterização dos modelos foram pesquisadas, extraídas e definidas com base em alguns trabalhos relacionados [20] [31] [55] e principalmente na bibliografia “Capacity Planning and Performance Modeling” [55] que pode servir às pessoas interessadas em aprofundar os estudos neste tópico.

3.2.1 Entendimento do Ambiente (E1)

A etapa de entendimento do ambiente em estudo é fundamental para o planejamento de todo o projeto de análise de desempenho. Um levantamento

de dados ou uma especificação equivocada do ambiente, causada por falta de conhecimento do ambiente ou interpretação incorreta, pode produzir reflexos nos resultados obtidos e impedir a obtenção de conclusões corretas.

É necessário compreender quais são os objetivos ou resultados esperados com o uso do ambiente em estudo e com o uso de servidores de w*eb-caching*. Sem esta expectativa fica mais difícil identificar o que deve ser analisado e praticamente impossível definir o que deve ser verificado.

É importante considerar e entender todos os parâmetros que influenciem no ambiente. Especificamente para a análise de desempenho de estruturas hierárquicas de servidores w*eb-caching*, é preciso considerar características de *hardware* e *software* em uso, rede física local (*LAN*) e de longa distância (*WAN*), canais de comunicação de dados entre diferentes servidores, carga de requisições, principais serviços Internet utilizados, características de clientes e servidores, protocolos de comunicação, enfim, todos os aspectos relevantes ao desempenho do sistema.

Todos os principais elementos que compõem o sistema hierárquico de w*eb-caching* devem ser estudados e analisados, por exemplo, área em disco, relações de cooperação e alternativas de recuperação dos objetos, assim como a interação entre os mesmos, seja qual for esta interação. Devemos entender o funcionamento do ambiente em partes, a relação entre as partes e como um todo.

Em alguns casos é relevante considerar questões de sazonalidade de utilização dos serviços Internet, por exemplo, época de férias, dia da semana e horário do dia. Estas, entre outras informações, servem para caracterização do Modelo de Carga. (E1.R12.M1).

O perfil dos usuários também pode influenciar na demanda dos serviços. Por exemplo, é esperado que em um ambiente acadêmico as consultas a *sites*

educacionais sejam maiores que em um provedor cujos usuários tem interesses distintos. Como mencionado no Capítulo 2, este perfil dos usuários pode representar 10% a mais de acertos nos servidores de w*eb-caching*, tornando-se um fator a ser ponderado no momento da escolha da estrutura hierárquica de w*eb-caching* da empresa [38].

3.2.2 Estudo e Escolha de Implementações de *Web-Caching* (E2)

São diversos os sistemas e as implementações de w*eb-caching* disponíveis para implantação de servidores w*eb-caching* e de estruturas hierárquicas de servidores w*eb-caching*. Algumas soluções utilizam *hardware* específico para essa função, outras não. Em comum todas apresentam um conjunto de *softwares* responsáveis em gerenciar recursos físicos que serão compartilhados por diversos usuários. Um exemplo é a administração de áreas de armazenamento de objetos que podem ser consultados novamente.

Algumas métricas de desempenho devem ser levadas em consideração ao se estudar e escolher uma implementação. Podemos citar duas com influência direta nesta decisão: volume total de requisições e de *bytes* a serem atendidos; e disponibilidade e confiabilidade dos recursos.

Os sistemas de w*eb-caching* possuem mecanismos distintos de cooperação. Alguns implementam protocolo padrão estabelecido por entidades internacionais. Outros implementam protocolos proprietários, normalmente mais eficientes quando a cooperação é estabelecida entre duas ou mais implementações iguais, mas podem piorar o desempenho quando a cooperação é realizada com outras implementações ou mesmo inviabilizar esta colaboração.

Para tornar a estrutura mais eficiente, é interessante considerar quais são os servidores de w*eb-caching* localizados mais próximos. Esta proximidade deve ser avaliada pelo tempo de resposta dos servidores distribuídos pela Internet

que permitem consultas originadas da rede em estudo, e se existe alguma limitação para cooperação entre os servidores. Estamos considerando, neste momento, a possibilidade de cooperação de servidores de diferentes instituições.

Definidas quais implementações serão usadas ou utilizando aquelas já em operação, precisamos avaliar todos os atributos de configuração que podem ser alterados, tais como tamanho de disco, quantidade de memória, controle de acesso, regras de substituição dos objetos armazenados, relação de cooperação entre os servidores e protocolo de cooperação. Estas informações são indispensáveis para alimentar os modelos de carga de trabalho (E2.R13.M1) e de desempenho (E2.R5.M2).

3.2.3 Definição de Parâmetros sob Estudo (E3)

São muitas as variáveis que influenciam no ambiente de servidores de w*eb-caching* e, dependendo dos objetivos e parâmetros que estarão sendo analisados, a possibilidade de um modelo analítico ser utilizado ou elaborado é muito pequena. Uma descrição de parâmetros normalmente analisados é apresentada no Capítulo 2.

Uma das variáveis mais complexas de ser analisada é a latência de resposta às requisições de objetos na Internet. Esta latência é influenciada, principalmente, pela concorrência nos meios de comunicação de dados da origem até o destino do endereço acessado, no retorno do objeto, no tamanho do objeto e no tempo de resposta do servidor. Porém existem outros componentes importantes na composição desta latência, por exemplo, tempo gasto desde a montagem da solicitação e posterior apresentação da informação por parte do cliente e tempo de decodificação e devolução do objeto no lado do servidor [59].

Como mencionado no Capítulo 2, algumas das métricas de maior interesse nos estudos de w*eb-caching* são taxa de acerto (*hit rate*) em número de requisições, taxa de acerto em volume de *bytes* e consumo de banda de *link* de comunicação de dados. Em relação aos parâmetros podemos citar espaço em disco para armazenagem dos objetos consultados, quantidade de memória RAM, relação de cooperação, e regras de substituição de objetos na área de *cache*. Estes não são os únicos, mas os que apresentam maior influência no comportamento dos servidores.

A taxa de acerto em número de requisições em um servidor de w*eb-caching* pode ser alta, mas se este acerto se concentrar em pequenos objetos a quantidade de *bytes* atendidos localmente não será tão representativa. Por outro lado se tivermos mais acertos em objetos grandes, a tendência é reduzir o número de acertos em requisições, significando que menos usuários estarão sendo beneficiados. A definição de um ponto de equilíbrio que satisfaça a todos os usuários é relativa e vai influenciar diretamente na política de substituição das informações armazenadas em disco e também em outros parâmetros. Neste momento, a satisfação será diferenciada para os grupos envolvidos. Esta característica terá influência na seleção da melhor estrutura (E10) representada pelo componente “Agentes” do método proposto. Portanto, é muito importante observar o impacto que a técnica de substituição de objetos em disco pode provocar no sistema como um todo e também se justifica a quantidade de estudos que consideram algoritmos de substituição do conteúdo em disco (Capítulo 2).

As políticas de substituição de objetos em disco são um assunto polêmico que foram discutidas em vários artigos, dissertações e teses, merecendo uma atenção especial por influenciar diretamente na base de dados que fica a disposição para recuperação local, conseqüentemente alterando o comportamento da métrica taxa de acerto e do parâmetro tempo de vida dos objetos em cache [62] [64].

A quantidade de memória RAM dos servidores de w*eb-caching* tem um impacto importante em todo o processo de atendimento das requisições dos usuários. Este dispositivo deve evitar ao máximo o acesso aos dispositivos mais lentos, como disco rígido. Em caso de *hit* na requisição, o tempo de acesso à memória RAM e ao disco rígido terão participação significativa na latência total do pedido do usuário. Em caso de *miss* no servidor de w*eb-caching* o tempo consumido por este dispositivo não terá muita participação no tempo total de resposta da consulta (na maioria dos casos), tendo em vista que o fator que causará maior atraso na resposta ao usuário será o tempo necessário para recuperar o objeto de outro servidor *web-caching* ou direto da origem do objeto.

A seleção dos parâmetros deve ser direcionada de acordo com os objetivos e resultados de desempenho esperados para o ambiente em estudo. Esta definição deve refletir de maneira decisiva na formulação dos modelos de carga de trabalho (E3.R14.M1) e de desempenho (E3.R6.M2).

Nesta etapa também é necessário definir os critérios de análise e comparação que serão aplicados aos resultados obtidos do sistema experimentado em laboratório e do sistema real. Estes critérios devem ser suficientes para garantir que o sistema simulado (E6) está com comportamento aceitável, ou seja, gerando resultados próximos dos recolhidos na etapa de coleta de dados (E5). Uma maneira de estabelecer estes critérios é definir margens de erros máximos aceitáveis para a taxa de acerto em requisições entre os sistemas e também erros máximos para outros possíveis parâmetros sob estudo.

3.2.4 Seleção de Aplicativos Auxiliares (E4)

Uma etapa necessária no método de análise de desempenho de estruturas hierárquicas de servidores w*eb-caching* é a seleção de aplicativos que auxiliem a extração de informações obtidas na coleta de dados (E5) durante os experimentos e na execução das simulações.

O grande volume de dados gerados e tratadas durante todas as etapas do método torna inviável a manipulação das informações sem o auxílio de aplicativos que automatizem alguns procedimentos.

Existem dois grupos de aplicativos que devem ser avaliados e selecionados. O primeiro tem como meta permitir a avaliação dos arquivos de registro de eventos (*log*) gerados nas simulações ou obtidos na coleta de dados. O segundo deve implementar as simulações no experimento em laboratório assumindo as funções de clientes e servidores *web* reais.

São apresentadas a seguir as descrições dos dois grupos de aplicativos auxiliares necessários para formulação dos modelos de carga de trabalho (E4.R15.M1) e modelo de desempenho (E4.R7.M2). Estas ferramentas também serão necessárias na fase de experimentação do método.

3.2.4.1 Seleção de Aplicativos de Análise de Eventos

Para realizar uma análise eficiente no ambiente sob estudo é preciso selecionar uma ou mais ferramentas que interpretem todos os arquivos de eventos e apresente as informações relevantes em formato claro e intuitivo.

Alguns programas para análise de arquivos de eventos são detalhistas em excesso e comprometem uma boa percepção do que está acontecendo no ambiente, confundindo o pesquisador e aumentando a possibilidade de falha na interpretação das informações.

Uma sugestão é determinar quais são as informações necessárias para avaliação dos parâmetros definidos anteriormente (E3) e então proceder com a avaliação de cada um dos aplicativos disponíveis, verificando qual apresenta a informação desejada de forma mais eficiente.

Em certos casos, vê-se necessário o desenvolvimento de programas particulares que extraiam dos arquivos de eventos exatamente o que se espera e que não foi oferecido por nenhum aplicativo disponível.

A instrumentação do código fonte do aplicativo servidor de w*eb-caching* é uma opção para gerar os dados que podem não constar dos arquivos de eventos originais do sistema w*eb-caching* em uso. Esta alternativa exige um pouco mais de experiência por parte do administrador do sistema ou de quem esteja realizando a análise, mas pode ser a única opção para se obter a informação necessária. Uma análise onde foi necessária a instrumentação do código fonte do sistema w*eb-caching* é apresentada em FONSECA [31].

A etapa de seleção de aplicativos é importante nas duas fases do método. Durante a caracterização e validação dos modelos, serve para fornecer subsídios às modelagens da carga de trabalho (E4.R15.M1) e de desempenho (E4.R7.M2) que serão utilizadas na fase de experimentação. Este subsídio é fornecido pela análise dos arquivos de eventos gerados no ambiente em produção ou de outra fonte escolhida.

O aplicativo de análise de arquivos de eventos também será importante durante as etapas de validação do modelo de carga (E6) e de desempenho (E7) onde os resultados extraídos dos arquivos de eventos permitem a validação dos modelos; na etapa simulação (E9), quando se torna necessário extrair o comportamento observado de uma estrutura após o término da carga sintética; na etapa de seleção da melhor estrutura (E10) quando são necessários resultados para efetuar comparações quantitativas; e por fim na etapa de validação em produção (E11) quando é necessário validar a estrutura escolhida em ambiente real.

3.2.4.2 Seleção de Aplicativos para Simulação

Assim como na escolha de aplicativos de análise de eventos, esta etapa exige uma avaliação adequada de diferentes aplicativos capazes de simular requisições de clientes e respostas de servidores. Além disso, é interessante que esta ferramenta seja capaz de gerar informações que auxiliem na avaliação dos parâmetros em estudos. Estas informações também podem ser obtidas pelos mesmos aplicativos de análise de eventos selecionados anteriormente.

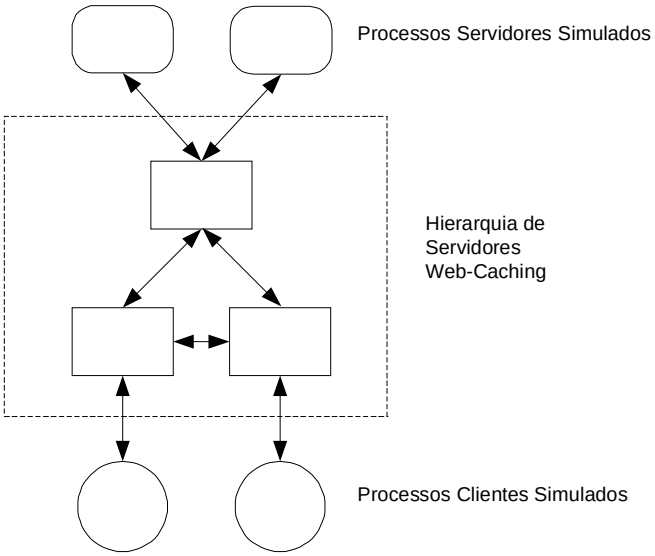
Normalmente, estes aplicativos (Capítulo 2) são compostos de dois grupos de processos. Um deles é formado por um ou mais processos clientes que disparam as requisições a um ou mais servidores de w*eb-caching*. O outro grupo de processos tem por finalidade simular servidores *web*, que atenderiam as requisições geradas pelos processos clientes. Na verdade as requisições estariam sendo repassadas ao servidor de w*eb-caching* que se encarregaria de armazenar uma cópia do objeto, evitando uma possível consulta futura ao mesmo conteúdo. A FIG. 3.2 apresenta o comportamento hipotético e genérico do ambiente simulado, onde a função dos processos “servidores simulados” e “clientes simulados” é exercida pelo aplicativo de simulação.

O aplicativo de simulação deve apresentar as seguintes características:

- Capacidade de gerar requisições aleatórias, ou obter tais requisições de outras fontes, por exemplo, arquivos de eventos;
- Permitir controle do número de requisições disparadas por segundo. Esta característica permite simular a existência de *links* de comunicação de dados mais lentos entre o Servidor de w*eb-caching* e o servidor *Web*, ou outro servidor de w*eb-caching* da estrutura hierárquica;
- Configurar o tamanho médio dos objetos requisitados;
- Controlar o compromisso de taxa de acerto em requisições x taxa de acerto em volume de dados;

- Permitir escolha do(s) servidor(es) w*eb-caching* para onde serão encaminhadas as consultas.

FIGURA 3.2 - Ambiente Simulado



Na FIG. 3.2 é possível observar a relação entre os três componentes básicos de um ambiente simulado. Os “processos clientes simulados” interagem com os servidores que compõem a estrutura hierárquica de servidores web-caching. Os servidores interagem entre si com o objetivo de verificar se o objeto requisitado já não se encontra dentro da arquitetura. Caso o objeto solicitado esteja dentro da arquitetura este é direcionado aos processos clientes, caso contrário a arquitetura se encarrega de recuperar o objeto dos “processos servidores simulados”.

Assim como o aplicativo de análise de eventos, o simulador é fundamental em diversas etapas do método proposto, entre eles podemos citar: validação e calibração do modelo de carga (E6) e de desempenho (E7), além da etapa de simulação (E9).

3.2.5 Coleta de Dados em Ambientes Reais (E5)

A coleta de dados de ambiente(s) real(is) tem o objetivo de obter o comportamento de acesso à Internet do ambiente sob estudo, permitindo uma formulação mais precisa dos modelos que serão utilizados durante a fase de experimentação.

As implementações de w*eb-caching* têm como característica em comum poder registrar grande quantidade de eventos fornecendo assim uma fonte de dados rica em informações que possibilite acompanhamento do ambiente e do seu desempenho.

É muito importante destacar a necessidade desta etapa. O responsável em realizá-la deve ter um grande comprometimento com todo o processo e estar ciente que qualquer falha nessa etapa do método implica no risco de geração de um modelo de carga de trabalho e de desempenho não representativo, que pode invalidar todo o processo de análise de desempenho.

Podemos identificar algumas possíveis maneiras de executar esta etapa: analisando um ambiente em produção na instituição em estudo; recolhendo dados de outras instituições equivalentes; utilizando informações disponibilizadas em outras análises; e gerando dados sintéticos em laboratório.

Quando a instituição possui um ambiente em produção, a coleta de dados pode ser realizada analisando os arquivos de eventos gerados pela implementação de w*eb-caching* em operação. Não é aconselhado que esta etapa seja ignorada, mesmo quando a instituição não possuir um ambiente de produção. Neste caso, o responsável em realizar a análise de desempenho deve proceder com a obtenção destes dados em outras instituições com as mesmas características de comportamento dos usuários ou em materiais publicados. Outra opção é a utilização de dados gerados a partir de simulações.

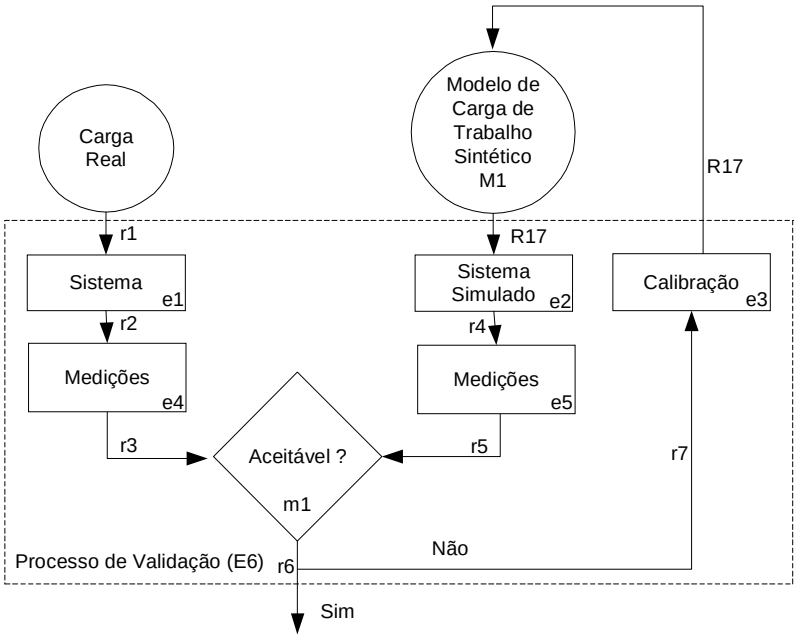
Todas as alternativas de coleta de dados devem ser bem avaliadas, evitando o aproveitamento de dados viciados por situações diversas, principalmente a opção que considera um comportamento hipotético.

Durante esta etapa é preciso: estimar a carga de trabalho que o sistema pode vir a sofrer; verificar a tendência de crescimento ou diminuição da carga a partir de dados históricos; analisar aspectos de desempenho; e analisar as estratégias, planos futuros e suas conseqüências na carga do sistema. Com isso prevêem possíveis mudanças no ambiente, que devem ser consideradas na formulação dos modelos de carga de trabalho (E5.R16.M1) e de desempenho (E5.R8.M2).

3.2.6. Validação e Calibração do Modelo de Carga (E6)

Esta etapa é uma das mais importantes do método proposto com influência direta no êxito de toda a análise de desempenho inicialmente planejada. Caso alguma falha seja cometida na validação e calibração do modelo de carga, toda experimentação e conseqüentemente as conclusões estarão sujeitas a erros.

FIGURA 3.3 - Validação e Calibração da Carga de Trabalho



A FIG. 3.3 apresenta o ciclo de ações necessárias para se alcançar um modelo de carga de trabalho sintético o mais representativo possível do ambiente real considerado. A carga de trabalho real, o modelo de carga de trabalho sintético, o sistema simulado, as medições, a aceitação ou validação, e a calibração são os componentes que formam esta etapa.

As Medições (e4) do Sistema real (e1) que é alimentado pela carga de trabalho real na verdade são os resultados recolhidos durante a etapa de coleta de dados de ambiente(s) real (is) (E5).

O modelo de carga sintético, nada mais é do que o Modelo de Carga de Trabalho (M1) formulado a partir de todas as informações levantadas nas etapas precedentes e que estará sofrendo ajustes até a finalização desta etapa.

O Sistema Simulado (e2) é a etapa de simulação do modelo de carga sintético (M1). Esta simulação consiste em preparar e executar a experimentação de um ambiente semelhante ao que originou os dados coletados anteriormente e que possui um comportamento conhecido, possibilitando comparação dos resultados. Ao preparar o ambiente simulado, deve-se tomar o cuidado de não deixar que eventos externos influenciem na simulação. Por exemplo, concorrência de acesso ao barramento de rede e pré-armazenamento em *cache*.

Em geral, precisamos garantir a limpeza completa do conteúdo da área de *cache* e do arquivo de eventos do servidor *web-caching*. Em seguida devemos promover um povoamento inicial desta área e prosseguir à simulação com o modelo de carga sintético definido.

As Medições (e5) são realizadas nos resultados obtidos durante a simulação no Sistema Simulado (e2), utilizando-se os aplicativos auxiliares selecionados (E4). Estas Medições (e5) devem formatar as informações geradas durante a

simulação de tal forma que facilite a comparação com os resultados recolhidos do ambiente real (e4.r3.m1).

Com os resultados das Medições de (e4) e (e5) e utilizando os critérios de comparações estabelecidos em (E3) podemos calcular os erros obtidos durante a simulação (m1). Caso os erros obtidos não permaneçam dentro dos limites pré-estabelecidos, deve-se proceder com a etapa de Calibração (e3) do Modelo de Carga Sintético (M1) e dar início novamente ao ciclo de ações.

A calibração do modelo de carga de trabalho ou modelo sintético (e3.R17.M1) é importante para aproximar o modelo formulado à situação encontrada no ambiente real. Esta etapa se torna necessária caso a validação/aceitação seja negativa depois de uma simulação. Sua função é ajustar o modelo de carga de trabalho sintético procurando remover os desvios que geraram as variações acima dos índices considerados aceitáveis.

O relacionamento (R17) é bidirecional entre a etapa de validação e calibração do Modelo de Carga (E6) e o Modelo de Carga de Trabalho (M1) pelo fato de existir interação entre etapa e o modelo em que o modelo alimenta a etapa com a carga de trabalho e outra interação onde a etapa ajusta o modelo até que a condição de aceitação seja atendida (m1).
É importante destacar que quando um problema no Modelo de Carga Sintético não for identificado durante esta etapa, o resultado desta falha, provavelmente, será observado apenas quando o processo de análise de desempenho alcançar a última etapa - a de validação em produção de todo o sistema (E11).

3.2.7 Validação e Calibração do Modelo de Desempenho (E7)

Assim como a etapa de validação e calibração do modelo de carga de trabalho (FIG. 3.3), esta etapa tem por finalidade ajustar o modelo de desempenho, verificando se as métricas definidas (tempo de resposta, taxa de acerto, *throughput*) para o modelo de desempenho convergem para as medidas

obtidas no ambiente real, permanecendo dentro de uma margem de erro aceitável (E7.R9.M2).

Esta validação também deve verificar se os resultados obtidos com o modelo de desempenho permitem análise quantitativa dos parâmetros realmente relacionados com as características importantes do sistema. A determinação do que é ou não importante na análise vai depender dos objetivos que foram definidos pelo pesquisador (E1). Por exemplo, se o objetivo do pesquisador é encontrar uma estrutura hierárquica de servidores de *web-caching* com alta taxa de acerto, o modelo de desempenho tem que explicitar esta característica nos resultados que serão apresentados.

3.2.8 Modelo de Carga de Trabalho (M1)

O Modelo de Carga de Trabalho ou Modelo Sintético, ou ainda modelo de *workload*, é uma representação da carga de trabalho real de um sistema sob estudo. Sua caracterização deve identificar os componentes básicos, escolher os parâmetros de caracterização, coletar dados e classificar os parâmetros [55].

O Modelo de Carga de Trabalho é a parte mais importante de qualquer projeto de análise de desempenho. As conclusões do estudo podem não ser aceitas caso a modelagem da carga seja inapropriada [71].

As quatro principais considerações no *workload* são: os serviços exercidos pelo *workload*, o nível de detalhamento, a representatividade e o "*timeliness*". Outras considerações menos importantes são: níveis de carregamento, impacto de outros componentes ou recursos e a recorrência dos objetos [71].

A formulação do modelo está condicionada a execução de várias etapas: entendimento do ambiente, estudo de sistemas de *web-caching*, definição de parâmetros sob estudo, seleção de aplicativos auxiliares e coleta de dados de

ambiente(s) real(is), além de estar sujeita a possíveis ajustes realizados durante a etapa de validação e calibração do modelo de carga (E6.R17.M1).

Em muitos casos, o Modelo de Carga de Trabalho é necessário devido ao grande volume de dados do ambiente real que inviabiliza trabalhar com dados empíricos de medições de desempenho do sistema. Quando se torna impraticável trabalhar com os dados empíricos, podemos substituir estas informações por uma representação mais compacta, gerando o Modelo de Carga de Trabalho simplificado. A adoção de um Modelo de Carga de Trabalho mais compacto inspira muito cuidado, esta decisão pode implicar em um risco que precisa ser considerado.

Este modelo deve ser representativo o suficiente para garantir comportamento semelhante ao que seria obtido com carga de trabalho real, mas isto não impede que simplificações sejam aplicadas à realidade observada.

A requisição é o componente principal da carga de trabalho de um servidor *web-caching*. Na verdade o conjunto de requisições, que devem ser classificadas de acordo com critérios que possibilitem caracterização da carga de trabalho de acordo com a aplicação em uso (ftp, *www*, vídeo), tamanho e probabilidade de acesso do objeto requisitado, e uma distribuição que determina a freqüência de requisições a um servidor Proxy.

No ambiente de estruturas hierárquicas de servidores *web-caching* é preciso considerar a carga de trabalho sendo gerada para vários servidores de *web-caching* distintos, dependendo do ambiente real, e induzir que os processos clientes gerem requisições com certo índice de repetição entre eles, possibilitando a cooperação entre os servidores. Se esta propriedade não for satisfeita a colaboração entre os servidores de *web-caching* tenderá a ser nula, comprometendo toda a análise. Portanto, é muito importante modelar os clientes sintéticos com o objetivo de provocar algumas requisições iguais aos

diversos servidores de *web-caching* gerando acerto entre os servidores que participam da estrutura hierárquica.

Caso a análise de desempenho seja realizada em uma rede local, deve-se ter o cuidado de simular a possível existência de *link's* de comunicação de dados que afetarão o desempenho do sistema. Por exemplo, se um servidor está interligado a outro passando por um *link* de comunicação de dados de 256 Kbps, este *link* poderá ser um ponto de contenção no tráfego entre os servidores, influenciando diretamente no tempo de respostas das requisições. Esta característica pode ser simulada variando o número de requisições por segundo que serão disparadas ao servidor, parâmetro que faz parte do modelo de carga de trabalho sintético.

A modelagem da carga de trabalho deve conter alguns parâmetros importantes, que podem ser definidos da seguinte forma [31]:

- Domínio: de acordo com o endereço de destino da requisição, distribui as requisições em conjuntos, cada conjunto pode ser atendido por um servidor *web-caching* distinto;
- Popularidade do domínio: percentagem de requisições médias disparadas para cada domínio;
- Área de *cache* x volume de dados em requisições: relação do tamanho de área de *cache* total disponível e do volume total de *bytes* das requisições disparadas periodicamente;
- Informações sobre objetos: tipo, distribuição, tamanho médio, recorrência, popularidade;
- Protocolo de transporte (por exemplo: TCP, UDP);
- Seqüência das requisições;
- Suporte simultâneo a diferentes aplicações.

O aplicativo auxiliar selecionado, que executa a simulação dos processos clientes e servidores *web*, deve ser capaz de manipular os parâmetros, citados

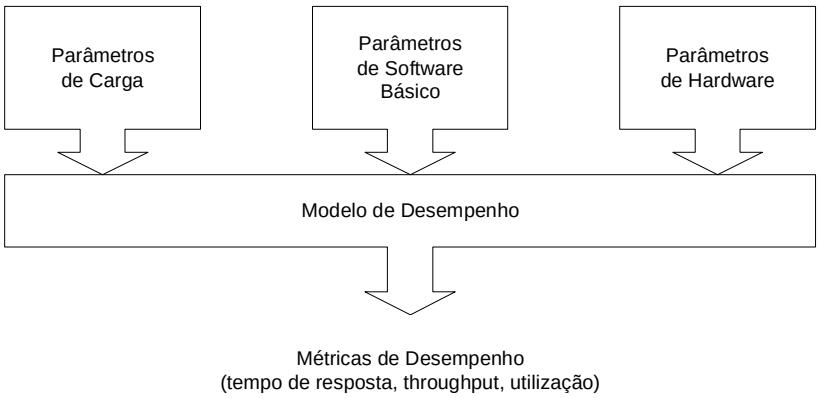
acima e também no Capítulo 2, que caracterizam a Carga de Trabalho no ambiente.

Na etapa de definição das estruturas hierárquicas a serem analisadas abordaremos as propriedades dos servidores de *web-caching* que são os elementos básicos sob estudos e que devem ser preliminarmente conhecidos.

3.2.9 Modelo de Desempenho (M2)

O modelo de desempenho é usado para estabelecer métricas de desempenho de um sistema Cliente/Servidor em função de três categorias de parâmetros: carga de trabalho, *software* básico e de parâmetros de *hardware* [55]. O resultado final desse modelo inclui previsões dos tempos de resposta do sistema, *throughput* (vazão), e utilização dos recursos do sistema, entre outros (FIG. 3.4).

FIGURA 3.4 - Representação gráfica do Modelo de Desempenho [55]



Os parâmetros de carga indicam a carga de trabalho ou volume de transações que serão submetidas ao sistema computacional. Estas informações são obtidas durante a caracterização do modelo de carga de trabalho na etapa de coleta de dados de ambientes reais (E5) e durante a definição de parâmetros sob estudo (E3). São informações do tipo: número de requisições diárias

atendidas pelos servidores de *web-caching*, tamanho médio de tráfego gerado por dia na instituição, tempo total de I/O dos servidores, etc.

O ambiente computacional contribui com mais duas informações para formulação do modelo de desempenho: os parâmetros de *software* básico e de *hardware* que compõem a estrutura hierárquica de *web-caching* em análise.

Os parâmetros de *software* básico fornecem informações do *software* básico utilizado no ambiente em análise, sendo elas: informações de sistema operacional e do *software* responsável pelo serviço de *web-caching*.

Os parâmetros de *hardware* como o próprio nome indica descrevem características do *hardware* do ambiente computacional. Além disso, deve considerar recursos de comunicação de dados, componente muito importante na análise de desempenho de aplicações que dependem de alta capacidade de transmissão e recepção no fluxo de dados.

3.2.10 Modelo de Custo (M3)

Não é escopo deste trabalho detalhar o modelo de custo associado a este método. Nos restringiremos a dizer que a função principal deste modelo é contabilizar os investimentos em *hardware*, *software*, recursos de telecomunicações, suporte, entre outros, permitindo estabelecer compromissos entre custo, benefício e desempenho das estruturas hierárquicas avaliadas.

Esse modelo pode e/ou deve ser definido e caracterizado em função do ambiente onde a estrutura está sendo avaliada e dos objetivos determinados com o uso dos servidores de *web-caching*.

3.3 Experimentação

Caracterizados e validados os modelos, partimos para a fase de experimentação, composta de: Definição (E8), Simulação (E9), Seleção da melhor (E10) e Validação de diferentes estruturas hierárquicas (E11) (FIG. 3.1). Em outras palavras, é a fase prática de aplicação dos modelos para fundamentação da análise de desempenho ou escolha da melhor opção de estrutura hierárquica para o ambiente.

É possível que durante a etapa de Simulação (E9), sejam necessários ajustes em um ou mais modelos estabelecidos (E9.R11.M2) (E9.R19.M1). Isto indica que a formulação e validação dos modelos não foram completas e esta falha pode acarretar problemas mais sérios que deverão ser observados na validação da estrutura selecionada (E11).

3.3.1 Definição de Estruturas Hierárquicas para Simulação (E8)

Os servidores de *web-caching* podem assumir algumas funções dentro de uma estrutura hierárquica de cooperação. As mais conhecidas são: *parent* (pai), *sibling* (irmão), *child* (filho) e *array* (idéia de agrupamento). As opções de cooperação variam de acordo com o sistema *web-caching* escolhido para implementação dos servidores de *web-caching*. Algumas implementam técnicas de colaboração padrões, outras utilizam técnicas proprietárias. As funções de cooperação disponíveis na implementação escolhida permitem direcionar a definição das estruturas a serem simuladas.

Uma requisição recebida por um servidor de *web-caching* será atendida por este servidor, independente da função deste dentro da estrutura hierárquica e se a consulta gerou ou não um *hit* no conteúdo já armazenado localmente. O importante é observar como o objeto será recuperado pelo servidor de *web-caching*, isto deve depender da função do servidor dentro da estrutura hierárquica definida.

Considerando que o sistema de *web-caching* usado por uma instituição para implantar servidores de *web-caching* é o Squid [97], podemos apresentar um cenário e explicar como poderia ser criada a estrutura de colaboração entre os servidores.

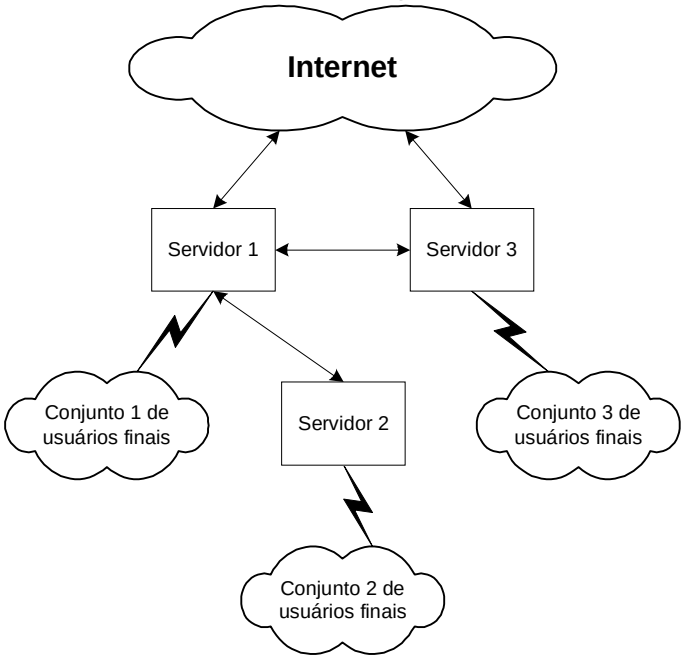
Os servidores que assumem a função de *parent* dentro de uma estrutura hierárquica, tem a responsabilidade de atender requisições geradas por outros servidores de *web-caching child*, mesmo que o material requisitado não esteja em sua área de *cache*.

Os servidores *sibling* atendem outros servidores *sibling,* apenas se a requisição recebida gerar um *hit* no conteúdo armazenado em sua área de *cache* local.
Por fim, os servidores *child* retransmitem as requisições recebidas dos usuários que causaram erro na base local do servidor para os servidores *parent* dentro da estrutura.

Todas as três categorias descritas acima têm a capacidade de atender a usuários finais. Em caso de não resposta em tempo hábil ou resposta negativa de todos os servidores dentro da estrutura hierárquica, qualquer um dos servidores de *web-caching* deve ser capaz de obter o objeto requisitado direto da origem. Ou seja, não obrigatoriamente toda requisição que gerar um *miss* em um servidor *child* será atendida por um servidor *parent*. Dentro de uma estrutura hierárquica, o servidor *parent* pode estar sobrecarregado ou fora de operação.

Na FIG. 3.5, o servidor 1 assume a função de *parent* para o servidor 2 e a função de *sibling* para o servidor 3. O servidor 2 é *child* do servidor 1, e o servidor 3 é *sibling* para o servidor 1. Cada um deles pode atender requisições de usuários finais.

FIGURA 3.5 - Estrutura Hierárquica de Web-Caching.



Outra característica comum é a configuração de servidores *parent* voltados para armazenagem de alguns domínios específicos, possibilitando balanceamento de carga [73]. Dessa forma pode-se atribuir a um servidor a responsabilidade de armazenar domínios ".br" e um outro ficaria com tudo que não fosse ".br", ou ainda, "!.br". Esta tentativa de balanceamento deve procurar dividir a carga de requisições pela metade, ou em uma proporção respeitando a capacidade dos servidores em questão.

Estas características possibilitam a definição de algumas propriedades dos servidores *web-caching* fundamentais para o sucesso da aplicação do método. Essas propriedades são [31]:

- Conectividade de pedidos: servidores para os quais o servidor requisitado pelo cliente pode retransmitir os pedidos, ou seja, quais são os servidores que assumem a função de pai e irmão para o servidor responsável em atender o cliente;
- Conectividade de resposta: quais são os servidores com permissão para requisitar colaboração, ou seja, quais são os servidores que assumem a função de servidores filhos ou irmãos do servidor colaborador;

- Cobertura de domínios: qual o domínio de endereços coberto pelo servidor.

Em ambientes compostos de vários servidores, a quantidade de diferentes estruturas hierárquicas possíveis de serem analisadas é enorme, tratando-se de um caso de combinação. Porém algumas orientações básicas reduzem este número consideravelmente.

Estas orientações levam em consideração a capacidade de cada servidor, a localização física e a largura de banda do canal de comunicação entre os servidores. Por exemplo, uma instituição com várias ramificações da rede em forma de estrela, com o ponto central da estrela na matriz, onde se encontra a saída para Internet, e as demais localidades interligadas com canais de comunicação de baixa capacidade, não justifica analisarmos soluções com servidores *parent* nas unidades remotas. O mais coerente é manter este servidor no centro da estrela. Ou ainda, eleger servidores *parent* aqueles o mais próximo possível do canal de escoamento do tráfego Internet.

Servidores de *web-caching* interligados com canais de comunicação de banda larga (por exemplo, um único barramento de rede), deveriam estabelecer cooperação como irmãos, desde que tenham capacidade de *hardware* para suportar a carga gerada.

Não é interessante mantermos servidores irmãos localizados em unidades distintas interligadas por canais de comunicação de baixa capacidade. Nestes casos, é mais indicado eleger um dos dois como pai e o outro como filho.

Considere que estamos analisando o impacto no somatório da taxa de acerto das requisições em função de variação na estrutura hierárquica de *web-caching*. É natural imaginarmos que quanto maior a relação entre os servidores, maior deve ser a taxa de acerto dentro da estrutura. Então é razoável trabalharmos com estruturas mais complexas de relacionamento com o objetivo de melhorar o parâmetro em análise.

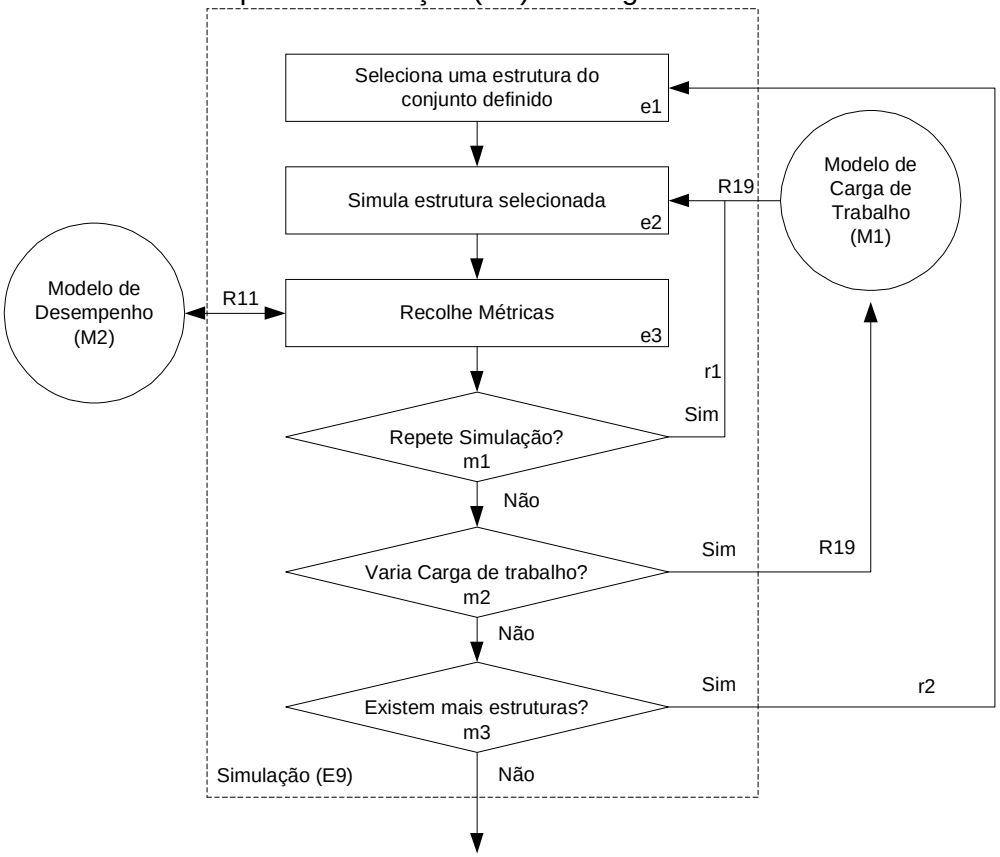
Em suma, escolhido o sistema de *web-caching* que será utilizado e verificando quais são suas características em relação aos tipos de colaboração que podem ser estabelecidas, é objetivo desta etapa propor diferentes estruturas hierárquicas, com implantações viáveis, que serão simuladas e sujeitas à análise de desempenho.

3.3.2 Simulação (E9)

A etapa de simulação tem por objetivo simular todas as Estruturas Hierárquicas definidas (E8), aplicando o modelo de carga de trabalho definido e validado (M1.R19.E9) e avaliando as métricas de desempenho definidas no modelo de desempenho (M2.R11.E9). Nesta simulação o ambiente laboratorial deve ser preparado tomando-se o cuidado de garantir condições semelhantes de execução para todos os experimentos e impedindo que componentes externos influenciem no comportamento dos servidores *web-caching*. Esta simulação é feita de partes do ambiente real usando ambiente experimental em laboratório.

É importante também montar as simulações respeitando proporcionalidade dos recursos disponíveis em ambiente real para o ambiente simulado, ou seja, realizar um planejamento operacional das simulações. Por exemplo, digamos que um dos servidores disponíveis no ambiente real tem área em disco de 20 GB. Podemos realizar a simulação com área de *cache* de 20 Mb e reduzir proporcionalmente os outros parâmetros de carga, por exemplo 1000 para 1. Outra relação pode ser estabelecida isto dependerá do planejamento realizado.

FIGURA 3.6 - Etapa de Simulação (E9). Fluxograma do ambiente simulado.



A FIG. 3.6 apresenta o fluxograma de atividades previstas para a simulação que formará massa crítica para tomada de decisão a ser realizada na penúltima etapa do método (E10) ou para análise de desempenho das estruturas hierárquicas.

É recomendado repetir os experimentos com o mesmo modelo de carga de trabalho e uma mesma estrutura (m1) para se estabilizar resultados com a obtenção de valores médios, seguindo o fluxograma (e1.e2.e3.m1.r1.e2). Terminado o ciclo de repetição deve-se, variar no mínimo uma característica (m2) e realizar mais uma série de simulações avaliando o comportamento das variáveis que compõem o modelo de desempenho (e1.e2.e3.m1.m2.R19.M1.R19.e2). Enquanto houver variáveis do modelo de carga a serem alteradas não devemos testar outra estrutura.

Então cada estrutura proposta pode ser simulada mais de uma vez para verificar o comportamento do servidor de *web-caching* com diferentes cargas de trabalho, em que alguns parâmetros do *workload* devem ser alterados (m2).

Finalizadas todas as repetições para estabilizar os resultados de uma estrutura hierárquica e todas as variações do modelo de carga de trabalho definida para análise, devemos alterar a estrutura hierárquica de servidores *web-caching* para outra especificada na etapa anterior, até que todas as estruturas definidas tenham sido simuladas (m3.r2.e1).

Ajustes nos modelos podem ser necessários durante a simulação (m2.R19.M1) (e3.R11.M2), porém esta não é uma tarefa trivial, tendo em vista que não é possível saber qual seria o exato comportamento da estrutura hierárquica em um ambiente real. Estes ajustes indicam falhas em etapas anteriores.

Se a distorção nos resultados não for perceptível, dificilmente serão detectados problemas no modelo de carga ou no de desempenho, e este desvio de comportamento só será verificado ao validar a estrutura selecionada (E11).

3.3.3 Seleção da Melhor Estrutura (E10)

Terminada a etapa de simulação e análise de todas as estruturas hierárquicas de *web-caching* definidas (E9), chega o momento de utilizar critérios que possibilitarão a seleção da melhor solução para o ambiente real. Estes critérios são baseados em: métricas do modelo de desempenho (M2.R23.E10); previsões de custos (E13.R4.E10); influência do componente “Agentes” (Agentes.R20.E10); e parâmetros do modelo de carga de trabalho (M1.R22.E10).

A análise das métricas definidas no modelo de desempenho contribuirá, em parte, para a seleção da solução mais eficiente, pois estará apresentando o comportamento dos parâmetros obtidos durante a simulação (M2.R23.E10).

É muito importante observar que o adjetivo “melhor” neste caso, não é necessariamente a que apresenta melhor desempenho, pois esta pode ser uma solução cujo custo x benefício não seja razoável. Este equilíbrio do custo x benefício depende do modelo de custo estabelecido (E13.R4.E10).

As diferentes demandas e expectativas dos agentes externos são uma característica que também pode influenciar na escolha da estrutura, não é comum encontrarmos referências desta natureza nos trabalhos realizados, normalmente apenas um propósito geral é considerado. Dedicamos um tópico exclusivo para discutir está participação dos agentes externos (Agentes.R20.E10).

Apresentados os argumentos acima, juntamente com informações do modelo de carga de trabalho (M1.R22.E10) e da análise de desempenho feita sobre os resultados das simulações, estão reunidas todas as informações necessárias para se decidir qual a estrutura hierárquica de servidores *web-caching* deve ser implantada no ambiente.

Devemos observar que o método pode ser usado para análise de desempenho ou para escolha da melhor estrutura para um ambiente. No primeiro caso a etapa E10 não é necessária.

3.3.4 Validação em Produção (E11)

A última etapa desse método prevê a validação em produção (ambiente real) dos resultados obtidos durante a fase de experimentação. Esta validação deve ser realizada com a estrutura hierárquica de servidores *web-caching* selecionada na etapa anterior ou com algumas estruturas no caso de análise de desempenho apenas. Não faz muito sentido pensar em validar todas as estruturas simuladas, devido ao tempo necessário para validar cada uma delas, no caso de escolha da melhor estrutura.

A validação consiste em colocar em operação a estrutura hierárquica selecionada ou em análise no ambiente computacional real e verificar se o comportamento e desempenho do ambiente, sob carga real da instituição, vai corresponder ao obtido durante a simulação (E9).

Um pequeno erro pode ser aceito, já que temos vários fatores que podem influenciar na carga de trabalho do sistema, fatores estes que podem não ter sido considerados ao se estabelecer o modelo de carga. Porém, se o comportamento for muito diferente do experimentado será preciso verificar quais foram os motivos que influenciaram nesta variação e retornar ao início do método para realizar os ajustes necessários e dar reinício a rotina de análise de desempenho (E11.R21.E1). Ou ainda, o uso de modelos simplificados em várias etapas pode gerar erros que invalidem os resultados da pesquisa e exigem novas modelagens.
A remodelagem da carga de trabalho do sistema ou de outros elementos utilizados na simulação será necessária até que o comportamento e desempenho da estrutura hierárquica de servidores *web-caching* no ambiente real seja semelhante ao obtido em simulações ou esteja dentro de uma variação aceitável.

Devemos salientar que é necessário algum tempo para se concluir algo em relação ao comportamento de uma estrutura hierárquica de servidores de *web-caching* em ambiente real. Este tempo deve variar de acordo com a instituição e sistema computacional sob análise. Portanto, é muito importante que o modelo de carga e de desempenho sejam bem calibrados logo no início do método para evitar retornos que, com certeza, causarão grandes atrasos no cronograma proposto.

3.4 Agentes

Denominamos como agentes em nosso método os diferentes grupos de pessoas que estão diretamente relacionados ao problema de análise de desempenho de uma estrutura hierárquica de *web-caching*. Podemos dividir estes grupos em: usuários, administradores técnicos do sistema e responsáveis administrativos.

Não é difícil imaginar uma certa discordância dos objetivos desses grupos, principalmente os usuários e os responsáveis administrativos. Vamos apresentar quais seriam as expectativas de cada um desses grupos e onde estão os possíveis conflitos.

Os usuários têm como objetivo, basicamente, baixa latência nas respostas das requisições geradas durante sua navegação. Além disso, esperam alta disponibilidade dos serviços solicitados e das informações pesquisadas. A maneira como esta qualidade de serviço está sendo atendida não é sua preocupação. Portanto o usuário não se interessa em saber se existe ou não uma complexa estrutura de recursos a sua disposição.

Os administradores técnicos procuram atender a demanda e a expectativa gerada pelos usuários, implantando soluções técnicas de acordo com os recursos disponíveis, que nem sempre são suficientes.

Os responsáveis administrativos procuram garantir a viabilidade financeira da organização, impedindo que recursos sejam adquiridos de forma desordenada e sem prévio planejamento, ou seja, sem preocupação com eficiência.

Considerando estas características apresentadas, podemos concluir que os usuários esperam o melhor desempenho possível, os administradores técnicos tentam garantir o melhor desempenho e os responsáveis administrativos procuram o menor custo e maior eficiência de operação para estrutura. Então

pode haver duas relações em questão, uma que trata custo x desempenho e outra de custo versus benefício que dependerá da visão específica de cada grupo citado.

Estas características podem ser conflitantes, pois na maioria dos casos, os investimentos em novos recursos estão diretamente associados à melhoria nos serviços. Para exemplificar, podemos considerar a constante demanda de atualização do *link* Internet, recurso caro, mas necessário na maioria dos casos para eliminar pontos de contenção de acesso às informações. Estes *upgrades* implicam em investimentos altos que em boa parte dos casos são vetados pela administração da organização.

3.5 Conclusões do Capítulo

Ao elaborarmos este capítulo nos preocupamos em apresentar um método mais específico para tratar o problema de análise de desempenho de estruturas hierárquicas de servidores de *web-caching*. Este método teve em grande parte suas etapas herdadas de métodos de análise de desempenho de estruturas Cliente/Servidor, e algumas outras etapas herdadas de métodos utilizados em alguns artigos e dissertações que avaliavam servidores de *web-caching* (Capítulo 2).

A execução de todas as etapas sugeridas em nosso método implica em uma grande dedicação de tempo e recursos. As etapas de Coleta de Dados (E5) e Validação em Produção (E11) podem durar semanas até que o comportamento da estrutura fique menos sujeito a comprometimentos causados por variações não esperadas durante o período. Por exemplo, em uma instituição de ensino a época de matrícula ou provas pode afetar o comportamento da estrutura hierárquica.

É comum encontrarmos em alguns casos restrições para disponibilização de recursos físicos, o que impede a execução adequada de todas as etapas

sugeridas no método descrito. Com isso, a fase de experimentação pode ficar prejudicada e o ambiente de laboratório pode não reproduzir de forma representativa a estrutura real. Neste caso, sugerimos a utilização de *softwares* ou a aplicação de artifícios que simulem a existência de equipamentos físicos que não existem. Por exemplo, para simular a existência de canais de comunicação que apresentem retenção de tráfego, podemos controlar a taxa de requisições que serão redirecionadas aos servidores.

Na impossibilidade de execução de todas as etapas previstas no método, acreditamos que algumas poderiam ser dispensadas, desde que o responsável pela análise esteja ciente das implicações que essa decisão pode provocar. Um método simplificado deve contar com pelo menos as seguintes etapas: Entendimento do Ambiente (E1); Definição de Parâmetros sob Estudo (E3); Seleção de Aplicativos Auxiliares (E4); Definição de Estruturas Hierárquicas para Simulação (E8); Simulação (E9); Seleção da Melhor Estrutura (E10). Além da necessidade de formulação dos modelos de carga de trabalho e de desempenho. A etapa de Coleta de Dados de Ambiente Real (E5) foi desconsiderada, apesar de sua importância, pois muitos aplicativos auxiliares (E4) já indicam modelos de carga de trabalho padrões como sugestão para simulações. Estes modelos de carga de trabalho padrões também permitem anular as etapas de validação dos modelos (E6 e E7) pelo fato de apresentarem modelos já calibrados.

É importante observar que o método proposto pode ser usado somente para análise de desempenho ou para escolha da melhor estrutura hierárquica com base na análise de desempenho, podendo existir ou não a obrigatoriedade da execução da etapa E10 e de mudanças na etapa E11 validando a estrutura hierárquica escolhida ou então algumas ou todas analisadas.

CAPÍTULO 4 – ESTUDO DE CASO – REDE ACADÊMICA DA PUC-MINAS

4.1. Introdução

Para verificar a eficiência do método de análise de desempenho de estruturas hierárquicas de servidores *web-caching,* proposto no capítulo anterior, aplicamos as etapas definidas no método em um ambiente onde era necessário determinar uma estrutura hierárquica com melhor desempenho entre várias alternativas possíveis de implementações.

Portanto, este capítulo tem como objetivo aplicar o método proposto em uma situação real, usando-o para escolher a estrutura com melhor desempenho no conjunto de estruturas consideradas e conseqüentemente verificando a eficiência do método proposto.

Selecionamos para o nosso estudo de caso a Rede Acadêmica da PUC-Minas. O principal aspecto deste ambiente que contribuiu para nossa decisão foi a demanda de serviços e acessos à Internet apresentada entre agosto de 1998 e dezembro de 2000. Neste período as necessidades de atualização do canal de comunicação para Internet foram praticamente semestrais.

Aproximando-se o final de cada período letivo, o canal de comunicação atingia ocupações plenas, transformando-se no ponto de contenção da rede e do sistema *web*, provocando insatisfação por parte de todos os usuários. Este é um dos problemas que servidores de *web-caching* procuram minimizar.

A familiaridade com essa rede de computadores, a liberdade de acesso ao sistema e a grande quantidade de alternativas de estruturas hierárquicas de servidores *web-caching* que poderiam ser adequadas ao ambiente foram outros aspectos que nos motivaram a selecionar a rede acadêmica da PUC-

Minas como estudo de caso. Além disso, podemos classificar entre média e alta a complexidade de interconexão da rede de computadores da PUC, característica esta que enriquece nossa pesquisa.

Uma das preocupações normais quando um recurso exige freqüentes alterações é verificar se este recurso está sendo utilizado para os fins planejados. Com o acesso irrestrito, torna-se difícil verificar que tipo de tráfego está ocupando os recursos de comunicação.

É comum encontrarmos esta mesma preocupação em outras organizações [72] [81] [97], inclusive em instituições não acadêmicas. Isto pode ser justificado por uma série de motivos, entre eles: queda de produtividade dos funcionários, elevação nos custos operacionais da empresa e planejamento.

Uma alternativa para solucionar o problema da rede acadêmica da PUC-Minas tinha que ser implantada, pois as atualizações constantes no canal de comunicação com a Internet, além de não serem uma solução que resolvia o problema de forma eficiente, elevava demasiadamente os custos de operação da rede e sempre estavam sujeitas a não aprovação. Outro aspecto preocupante era que a falta de controle do conteúdo acessado pelos usuários estava acarretando utilização incorreta dos recursos.

A solução proposta foi a implantação de servidores de w*eb-caching*, distribuídos em pontos estratégicos da rede exercendo três funções básicas:

- Redução no uso de recursos: armazenagem local de objetos da Internet evitando consultas futuras ao *site* de origem, reduzindo consumo de canal de comunicação para Internet e tempo de resposta ao usuário;
- Controle de acesso de acordo com usuários, local ou horário: não são todos os usuários que devem ter acesso à Internet sem restrições, por exemplo, durante provas em laboratório não é desejado o acesso à rede;

- Controle de Conteúdo: restrição de acesso a alguns *sites* considerados de conteúdo impróprio.

Apresentada a motivação que nos direcionou para escolha do ambiente de nosso estudo de caso, partimos para aplicação da fase de caracterização e validação dos modelos e posteriormente da fase de experimentação definidas no método proposto no Capítulo 3.

4.2 Caracterização e Validação dos Modelos

4.2.1 Entendimento do Ambiente (E1)

A disposição geográfica da rede acadêmica de computadores da PUC-Minas segue uma topologia estrela, na qual todos os *campi* que compõem a Universidade se interligam ao campus principal, localizado na unidade BH - Coração Eucarístico (CE) que, por sua vez, mantém o circuito de acesso à Internet.

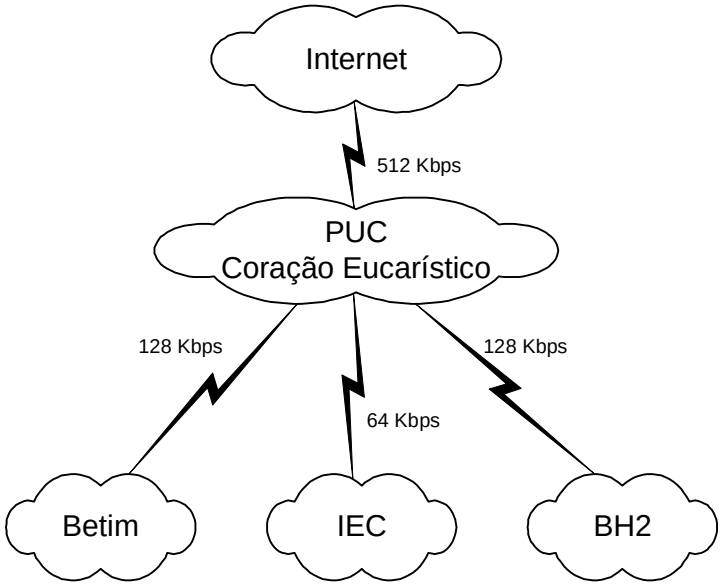
Durante o estudo de caso, não consideraremos os *campi* de Arcos e Poços de Caldas. Estas unidades mantêm conexão à Internet independentes e a administração desses *campi* é realizada por equipes técnicas distintas.

Outras considerações merecem comentários:

- Os usuários atendidos nos *campi* possuem um perfil semelhante, o que enquadra nosso ambiente dentre aqueles que podem apresentar um ganho adicional em possuir usuários com interesse em comum [38];
- Cada unidade, interligada ao campus principal, possui em torno de 110 estações clientes que utilizam o servidor de *web-caching* da unidade;
- A unidade Coração Eucarístico conta com um parque de 600 computadores, distribuídos entre laboratórios, salas de aulas, salas de professores, diretórios acadêmicos, e diretório central de estudantes;

- A sazonalidade é uma característica importante. Durante o final do semestre letivo a demanda por recursos de Internet aumenta. Épocas de provas, eventos e congressos, são outros exemplos de sazonalidade que afetam o comportamento dos usuários.

FIGURA 4.1 -Disposição física da rede acadêmica da PUC-Minas durante o período de análise do Estudo de Caso.

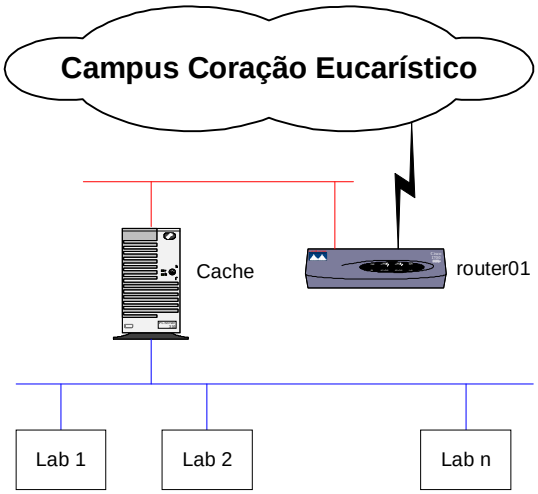


Como pode ser observado na FIG. 4.1, durante o período de análise do Estudo de Caso, as unidades estavam interligadas ao campus principal por canais de comunicação de dados de 128 Kbps, exceto o IEC que possuía um canal de 64 Kbps. O canal de comunicação para Internet estava limitado a uma taxa de transferência de 512 Kbps. Nas vésperas da defesa da dissertação apenas o canal de comunicação do campus Coração Eucarístico para Internet havia sofrido alteração passando de 512 Kbps para 2 Mbps.

Física e logicamente as redes de todos os *campi* possuem as mesmas características. Com duas placas de rede, um servidor isola a rede interna formada principalmente de laboratórios e a rede onde se encontra o roteador, assumindo as funções de *firewall* e *web-caching*. Esta separação física garante

que a navegação na Internet seja feita única e exclusivamente passando por este servidor (FIG. 4.2)

FIGURA 4.2 - Estrutura física e lógica das redes dos *campi* IEC, Betim e BH2.



Alguns laboratórios estão conectados à rede em *switchs* outros em *hubs* com taxas de transferência de 10 ou 100 Mbps. Na maioria dos casos, esta heterogeneidade não deve causar impacto na recuperação dos objetos requisitados pelos usuários. Podemos afirmar isto tendo em vista que o ponto de contenção da comunicação de dados não são as redes locais e sim os *links* que interligam as unidades (indicado na FIG. 4.2 pela ligação entre o router01 e a nuvem Campus Coração Eucarístico) e o canal de comunicação de dados para a Internet.

No caso do campus Coração Eucarístico, os servidores Cache PUC e DCC, descritos a seguir, estão interligados a 100 Mbps em *switch*s, onde também estão os roteadores que encaminham os dados para as outras unidades e para a Internet.

A seguir descrevemos as características técnicas de cada um dos servidores que formam o ambiente. Também estaremos atribuindo designações a cada um deles para facilitar referências futuras.

O campus Coração Eucarístico possui quatro servidores de w*eb-caching*, porém consideramos apenas dois durante nossas experimentações, sendo eles:

1. Servidor w*eb-caching* principal:
   - Designação: Cache PUC;
   - Domínio Internet: cache.pucmg.br;
   - Máquinas atendidas: aproximadamente 350 distribuídas dentro do campus principal;
   - Características técnicas: Pentium III, 700 MHz, 512Mb RAM, 14 Gb de área para *cache* de *web-caching*.

2. Servidor w*eb-caching* do Departamento de Ciência da Computação:
   - Designação: Cache DCC;
   - Domínio Internet: cache.dcc.pucmg.br;
   - Máquinas atendidas: 140 máquinas distribuídas em 11 laboratórios;
   - Características técnicas: Pentium III, 450 MHz, 512Mb RAM, 5 Gb de área para *cache* de *web-caching*.

Em cada uma das outras unidades estudadas contamos com apenas um servidor, cujas características descreveremos abaixo:

3. Servidor w*eb-caching* de Betim;
   - Designação: Cache Betim;
   - Domínio Internet: cache.betim.pucmg.br;
   - Máquinas atendidas: aproximadamente 110 máquinas distribuídas em 6 laboratórios;
   - Características técnicas: Pentium II, 166 MHz, 128 Mb RAM, 1 Gb de área para *cache* de *web-caching*.

4. Servidor w*eb-caching* de IEC;
   - Designação: Cache Iec;

- Domínio Internet: cache.iec.pucmg.br;
- Máquinas atendidas: aproximadamente 110 máquinas distribuídas em 6 laboratórios;
- Características técnicas: Pentium II, 133 MHz, 64 Mb RAM, 0.7 Gb de área para *cache* de *web-caching*.

5. Servidor w*eb-caching* do Campus São Gabriel ou BH2

- Designação: Cache BH2;
- Domínio Internet: cache.bh2.pucmg.br;
- Máquinas atendidas: aproximadamente 100 máquinas distribuídas em 6 laboratórios;
- Características técnicas: Pentium III, 450 MHz, 512 Mb RAM, 3 Gb de área para *cache* de *web-caching*.

Nesse momento, também é necessário identificar a participação do componente “Agente” definido no método proposto. Em nosso estudo de caso é possível verificar anseio muito grande por parte dos usuários em acessar os mais diferentes objetos na Internet, mesmo aqueles não relacionados as suas atividades acadêmicas, por este motivo é praticamente inviável atender esta expectativa. O interesse dos responsáveis pela instituição é garantir condições necessárias para um perfeito andamento das disciplinas e dos cursos, tornando a Internet uma fonte de material adicional para os alunos. O administrador de rede, por sua vez, precisa filtrar acessos impróprios que degradam o acesso a materiais realmente de cunho acadêmico, ao mesmo tempo atendendo a expectativa dos responsáveis pela instituição e dos usuários com interesses acadêmicos.

Esta etapa é encerrada atingindo os objetivos descritos no método. O ambiente computacional foi identificado em detalhes, as informações necessárias foram recolhidas e a análise de desempenho pode seguir para as próximas etapas.

4.2.2 Estudo e Escolha de Implementações de *Web-Caching* (E2)

Para o estudo de caso da PUC, foram verificadas duas alternativas viáveis para implantação de servidores w*eb-caching*. Uma utilizando um *software* gratuito que executa sobre plataforma UNIX/Linux e outra de custo razoavelmente baixo, que tem como requisito plataforma Windows NT, respectivamente Squid [97] e Microsoft Proxy Server [58].

Esta limitação de alternativas se deve à disponibilidade de recursos e a urgência em se implantar uma solução para reduzir o consumo de banda Internet da instituição. Outras soluções foram verificadas, porém estas necessitavam de aquisição de novo hardware, o que foi vetado. Além disso, as duas soluções atendiam a demanda existente no ambiente de forma bastante satisfatória.

Após algumas avaliações em ambiente real foi possível identificar as principais características de cada uma das soluções oferecidas pelos *softwares* selecionados. O sistema de *web-caching* Squid foi escolhido por apresentar algumas vantagens sobre o Microsoft Proxy Server. Entre elas:

- Capacidade de gerenciamento remoto mais eficiente, que permite suporte a distância evitando deslocamentos entre as unidades;
- Regras de colaboração mais amplas, possibilitando diferentes arranjos na estrutura hierárquica de servidores;
- Controle de acesso mais eficaz, tanto por origem como destino das requisições;
- Alterações das configurações com servidores on-line, aplicando simples reinicialização do serviço, enquanto que o Microsoft Proxy Server, em determinadas situações, necessitava recarga de todo o sistema operacional;
- Maior quantidade de parâmetros para configuração com efeitos diretos no desempenho do sistema, por exemplo, *timeout* de acesso entre

servidores cooperados, seleção da regra de substituição do conteúdo do *cache* e controle de acesso por palavra chave;

- Facilidade de suporte, tendo em vista sua grande aceitação na Internet;
- Solução amplamente utilizada e na maioria das vezes indicada por vários administradores de rede;
- Código fonte aberto o que permite alterações ou instrumentações no *software*, adaptando-o à situação da instituição.

Esta etapa alcançou seus objetivos, foi escolhido o sistema de *web-caching* que será utilizado e suas características foram avaliadas, permitindo continuar a análise de desempenho.

4.2.3 Definição de Parâmetros sob Estudo (E3)

Para definirmos quais são os parâmetros do sistema e da carga de trabalho que estaremos acompanhando durante as experimentações devemos levar em consideração quais são os principais objetivos da instituição, ou ainda, o que se espera com a implantação de uma solução com servidores de *web-caching*. Os problemas a serem solucionados indicam os parâmetros.

Em nosso caso, o objetivo principal é reduzir a necessidade de redimensionamento nos canais de comunicação e reduzir o tempo de latência na resposta às requisições dos usuários. Para tanto, precisamos elevar ao máximo a taxa de acerto nas requisições formuladas pelos usuários, isto implica não só em uma elevada taxa de acerto em números de requisições, mas também em uma elevada taxa de acerto em *bytes* reaproveitados. Por exemplo, podemos ter 60% de acerto de número de requisições, porém isto pode representar apenas 15% de todo tráfego em *bytes* gerados.

O desejado é encontrar um ponto de equilíbrio entre taxa de acerto em número de requisições e quantidade de *bytes* trafegados. Porém isto pode nos levar a

uma série de discussões que sairão do escopo de nosso estudo de caso que procura comprovar a eficiência do método proposto.

Voltando ao objetivo delineado, podemos associar parâmetros como relação de cooperação entre os servidores, taxa de acerto em número de requisições e taxa de acerto em volume de dados. A relação de cooperação entre servidores quando alterada produz efeitos na forma como os servidores devem recuperar os objetos que causaram erro na base *web-cache* local. O volume total de bytes que serão recuperados direto da origem dos objetos requisitados será o parâmetro que permitirá comparação quantitativa entre as estruturas hierárquicas sob análise.

Poderíamos considerar outros parâmetros como tamanho de disco ou canal de comunicação. Porém isto elevaria consideravelmente a complexidade dos ensaios planejados e aumentaria o tempo necessário para realizar todas as simulações. Portanto, resolvemos realizar experimentações suficientes o bastante para comprovar a eficiência do método proposto no trabalho.

A etapa foi realiza, permitindo prosseguir na análise de desempenho, tendo em vista que foram determinados os parâmetros que serão avaliados - taxa de acerto em quantidade e taxa de acerto em volume de dados.

4.2.4 Seleção de Aplicativos Auxiliares (E4)

Avaliamos ferramentas de análise de arquivos de eventos específicas para o sistema de *web-caching* escolhido em nosso estudo de caso, ou seja, o *software* Squid. A pesquisa começou com informações e orientações obtidas no *site* oficial que distribui a ferramenta, mas também avaliamos aplicativos disponibilizados por outras organizações.

Após o estudo de algumas alternativas, conseguimos identificar as características das ferramentas disponíveis e escolher a que aparentemente

melhor se adequava ao nosso estudo de caso. Devemos lembrar que a escolha levou em consideração o formato em que os resultados eram apresentados e quais eram os parâmetros que se evidenciavam nos relatórios gerados.

Como ferramenta para análise de eventos avaliamos o aplicativo que compõem o próprio *software* Squid [97] denominado “Cache Manager” além dos aplicativos “Calamaris” [11], “Squid Log Analizer” [83], “Squid Manager Log” [97], “Squid Times” [89] e mais alguns menos interessantes que não chegaram a se enquadrar dentro das necessidades definidas.

O aplicativo de análise que acompanha o Squid, Cache Manager, gera relatórios complexos e de difícil entendimento. O “Squid Manager Log”, não produz relatórios claros, mas possibilita extrair informações de distribuição dos domínios Internet acessados, ou seja, qual a percentagem de acesso para domínios “.br”, “.com.br”, etc. Esta ferramenta é interessante para os responsáveis em executar a análise de desempenho que irão considerar hierarquias de servidores de w*eb-caching* com distribuição de domínios.

O aplicativo “Calamaris” se destacou por ser prático e objetivo**.** As informações mais importantes obtidas com este aplicativo são a taxa de acerto em número de requisições (quantidade e percentagem) e a quantidade de *bytes* (em *kilobytes* e percentagem). Além disso, informa como o servidor w*eb-caching* atendeu requisições que geraram erro na base local, sendo três as possibilidades: direto do servidor origem; com acerto no servidor pai e/ou irmão; e com o servidor pai recuperando o objeto.

Em nosso estudo de caso, o “Calamaris” se apresentou mais eficiente para alimentar a formulação dos modelos e para acompanhar o comportamento das estruturas hierárquicas que serão simuladas durante as experimentações. Por este motivo esta foi a ferramenta escolhida para analisar os arquivos de eventos.

Em relação à ferramenta de simulação de clientes e servidores de *web*, avaliamos os aplicativos “Web Cache Simulator” [17] e o “Web-Polygraph” [79]. Este último se adequou melhor as características das simulações planejadas, permitindo melhor manipulação no comportamento dos Robôs que assumem as funções de clientes, requisitando conteúdo *web* através dos servidores de *web-caching*.

Com o “Web-Polygraph” é possível: especificar os endereços de processos clientes, servidores *web* e de servidores *web-caching*; determinar os tipos de objetos que serão requisitados e suas propriedades; determinar a distribuição dos objetos; especificar quantidade de processos clientes com recorrência de cada um ou entre eles; e outras características que são mencionadas com exemplos no modelo de carga de trabalho (M1).

Simulações preliminares foram executadas com a ferramenta e não encontramos dificuldades em simular o comportamento que prevíamos ser necessário na etapa de validação e calibração do modelo de carga (E6) e na etapa de simulação (E9).

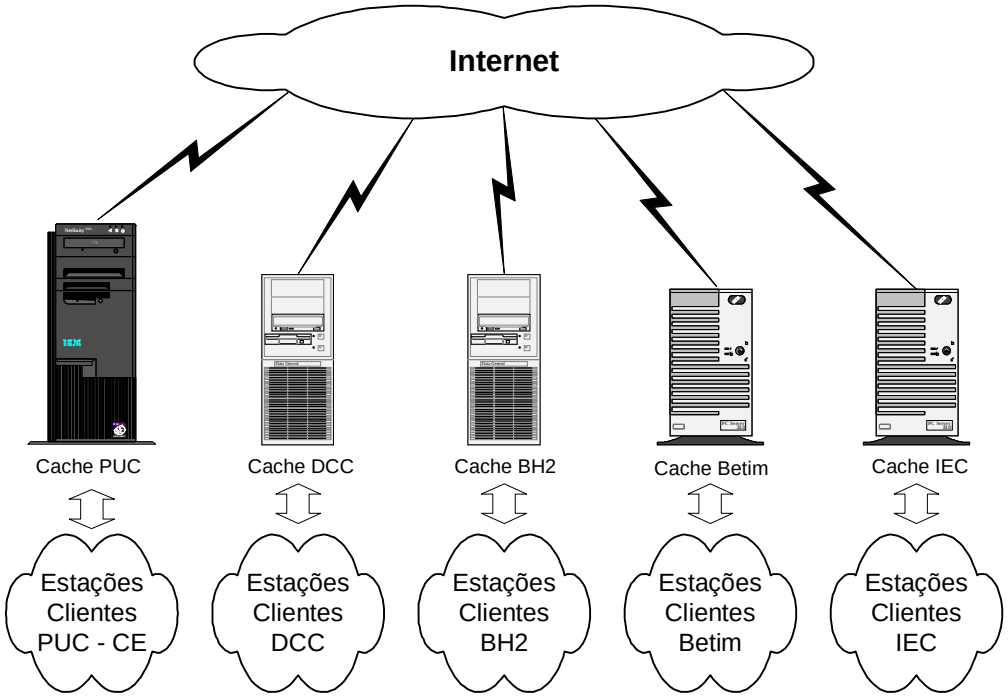
A etapa foi concluída com a seleção realizada dentro das necessidades do estudo de caso, o que valida mais uma etapa do método proposto.

4.2.5 Coleta de Dados de Ambiente(s) Real(is) (E5)

Esta foi uma etapa muito demorada, pois planejamos acumular durante aproximadamente três meses os arquivos de eventos de cada um dos servidores descritos na etapa de “entendimento do ambiente” (E1). Fomos um pouco mais além. Resolvemos aplicar uma alteração na estrutura hierárquica de servidores e acompanhar o comportamento do sistema em produção com carga real, de forma a nos auxiliar em etapas futuras do método.

Recolhemos os arquivos de eventos dos servidores que estavam dispostos sem nenhum tipo de colaboração entre eles, ou seja, nenhum deles trabalhava cooperando com outro servidor na rede acadêmica da PUC. Toda requisição não atendida localmente pelo servidor requisitado gerava uma consulta direta à origem do endereço Internet pesquisado. Esta estrutura (FIG. 4.3) esteve em operação durante mais de um ano, porém recolhemos seus resultados apenas durante oito semanas.

FIGURA 4.3 - Estrutura 1: Disposição lógica do acesso à Internet de cada um dos servidores *web-caching* sem hierarquia, em análise na coleta de dados.



A FIG. 4.3 não está apresentando exatamente a disposição física dos servidores, mas sim a disposição lógica, evidenciando a capacidade de cada um dos servidores atingir diretamente a Internet sem passar por nenhum outro servidor de *web-caching*. Na realidade, os servidores Cache IEC, Betim e BH2 trafegam seu sinal até a rede do *campus* Corações Eucarístico (PUC-CE) (FIG. 4.1). Este tráfego entra em disputa pela utilização do canal de comunicação de dados para Internet com as requisições dos servidores Cache PUC e Cache

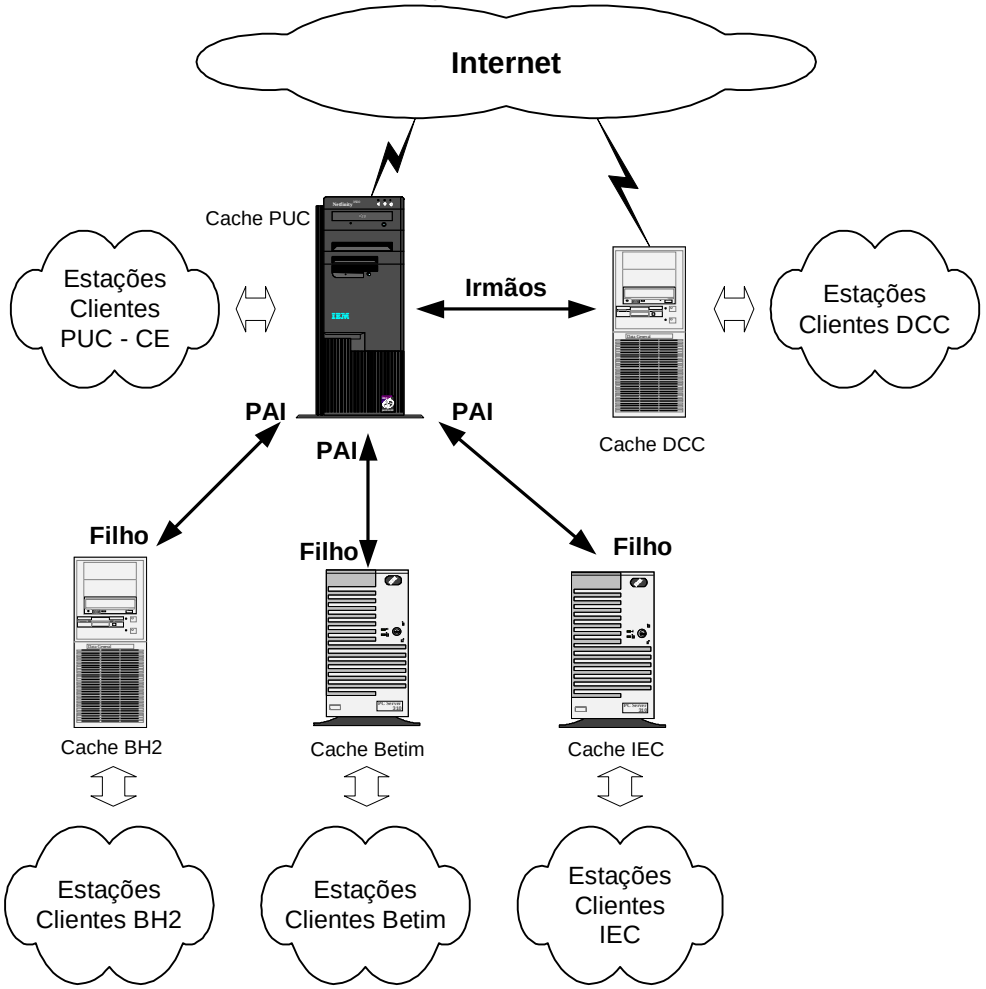
DCC, além de outros serviços que possam estar sendo disponibilizados na rede.

Decorridas as oito semanas, fizemos uma alteração na relação de cooperação de alguns dos servidores. Os servidores Cache DCC e Cache PUC trabalharam sendo um irmão do outro, ou seja, quando uma consulta não era atendida localmente, uma consulta era gerada para o servidor irmão. Se este tivesse o conteúdo solicitado atendia, caso contrário, não fazia nada. É importante observar que estes dois servidores se encontram no campus principal, Coração Eucarístico, tendo em comum acesso à Internet com o mesmo desempenho (FIG. 4.4).

O Servidor Cache PUC, além de ser irmão do Cache DCC, assumiu a função de pai para os outros três servidores sob análise: Cache IEC, Betim e BH2. Ou seja, o Servidor Cache PUC, recebendo uma requisição de um dos servidores filhos, mesmo não tendo o conteúdo solicitado localmente, se encarregava de recuperar a informação na Internet. Esta característica (do servidor Cache PUC) de recuperar a informação para o filho nem sempre é efetuada, pois o servidor filho trabalha com uma variável que determina quanto tempo deve ser aguardada uma resposta do pai antes de tentar recuperar o objeto direto da origem. Esta condição de desvio é importante para não retardar a resposta ao usuário nos momentos em que o servidor pai esteja sobrecarregado ou fora de operação (FIG. 4.4).

É importante observar que a FIG. 4.4 não mostra a capacidade dos servidores Cache IEC, Betim e BH2 de acessar à Internet diretamente caso o tempo de vida da requisição enviada ao servidor Cache PUC expire. Esta possibilidade não foi representada na figura, pois poderia confundir o leitor e dar uma falsa impressão do que realmente acontece.

FIGURA 4.4 - Estrutura 2. Disposição lógica da estrutura hierárquica de Servidores w*eb-caching* avaliada no estudo de caso.



Apresentadas as estruturas utilizadas para coleta de dados do ambiente real, mostraremos os resultados de cada um dos servidores observados, tecendo comentários sobre determinados valores obtidos.

Antes de qualquer análise dos valores obtidos, esperávamos uma uniformidade nos resultados dos servidores *cache* DCC, IEC, Betim e BH2 em ambas as estruturas. Estes servidores, apesar da mudança na estrutura, só deveriam apresentar alterações nos resultados das consultas que geraram erro no *web-cache* local, pois agora eles tinham novas alternativas para recuperação das requisições. Na Estrutura 1 estes servidores sempre obtinham o objeto do

servidor origem, na Estrutura 2 eles contavam com o reaproveitamento de informação armazenadas no servidor pai ou no irmão do pai.

Estruturamos este tópico apresentando o comportamento de cada um dos servidores *web-caching* monitorados e depois os resultados coletivos juntamente com uma análise geral da coleta de dados.

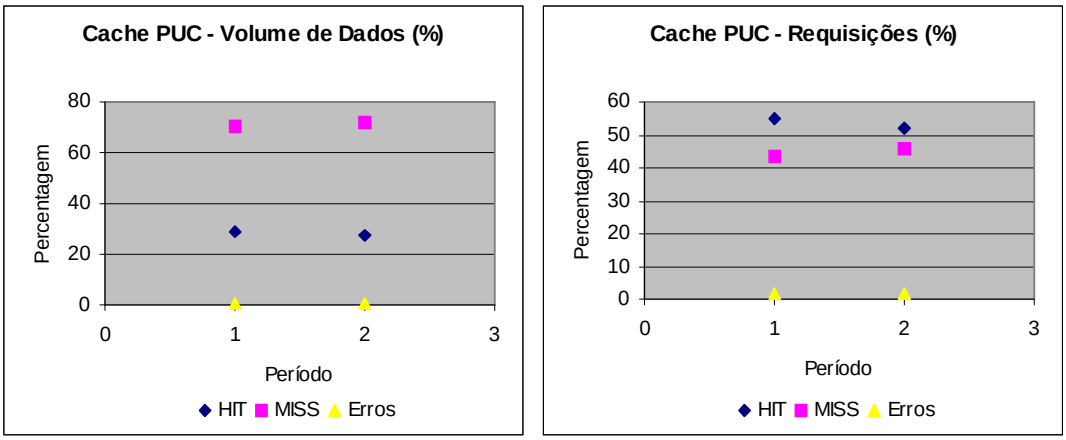
Durante a etapa de coleta de dados alguns arquivos de eventos foram perdidos ou descartados. Podemos descrever os seguintes problemas que causaram tais perdas: tamanho dos arquivos de eventos que dificultavam sua manipulação; arquivos apagados, por pessoas não orientadas ou pelo próprio aplicativo, para liberação de espaço nos servidores; arquivos corrompidos; semanas de manutenção técnica no servidor. De qualquer forma os poucos arquivos perdidos não invalidam a etapa ou os resultados coletados, tendo em vista que o comportamento padrão dos servidores pode ser avaliado.

As FIG. 4.5 a 4.9 apresentam os resultados coletados em percentagens de *HIT* (acerto), *MISS* (falha) e Erros das quantidades de requisições e volume de dados enviados a cada um dos servidores de web-caching. Cada ponto (losango, quadrado e triângulo) representa a média aritmética observada dentro de cada um dos períodos identificados no eixo x (Períodos). Os resultados apresentados contam apenas com valores obtidos nas consultas ao *web-caching* local de cada servidor, portanto, um acerto de um servidor filho no servidor pai não é calculado como acerto no servidor filho, é calculado acerto apenas no servidor pai.

4.2.5.1. Resultados Cache PUC

Mantivemos os arquivos de eventos do servidor Cache PUC entre os dias 11 de outubro e 4 de dezembro de 2001. Os resultados coletados entre o dia 11/10 e 11/11 estiveram sobre a Estrutura 1 apresentada na FIG. 4.3 e seu resultado médio é apresentado no Período 1 da FIG. 4.5. Entre os dias 11/11 e 04/12 os servidores estavam dispostos de acordo com a Estrutura 2 apresentada na FIG. 4.4 e seu resultado médio é apresentado no Período 2 da FIG. 4.5.

FIGURA 4.5 - Distribuição dos resultados do Servidor Cache PUC.

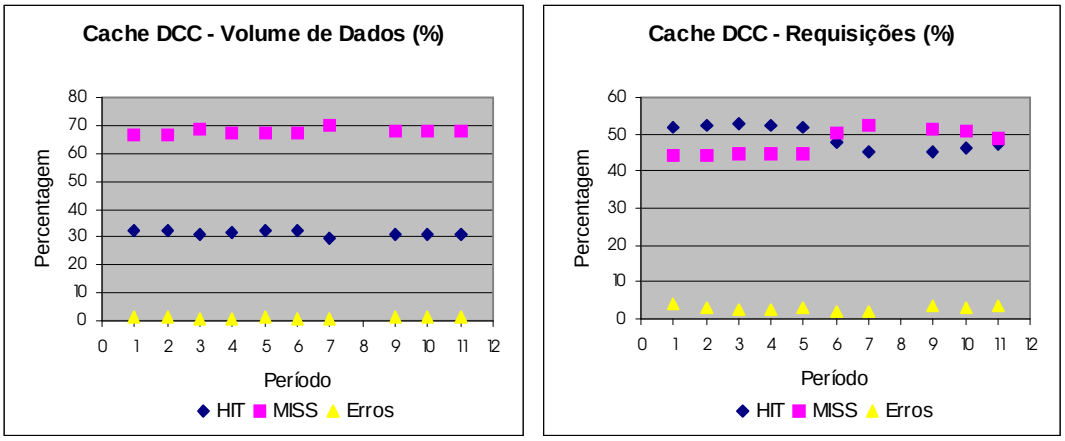


4.2.5.2 Resultados Cache DCC

Mantivemos os arquivos de eventos do servidor Cache DCC entre os dias 09 de setembro e 02 de dezembro de 2001. Os resultados obtidos entre o dia 09/09 e 04/11 estiveram sobre a Estrutura 1 apresentada na FIG. 4.3 e seus resultados médios semanais são apresentados do Período 1 a 8 na Fig. 4.6. Entre os dias 11/11 e 02/12 os servidores estavam dispostos de acordo com a Estrutura 2 apresentada na FIG. 4.4 e seus resultados médios semanais são apresentados do Período 9 a 11 na Fig. 4.6.

A razão de trabalharmos com períodos de resultados médios diferentes durante a coleta de dados dos servidores *web-caching* se deve a característica dos arquivos de eventos de cada um desses servidores. A maioria dos servidores renovava os arquivos de eventos semanalmente e outros tinham períodos maiores.

FIGURA 4.6 - Distribuição dos resultados do Servidor Cache DCC.

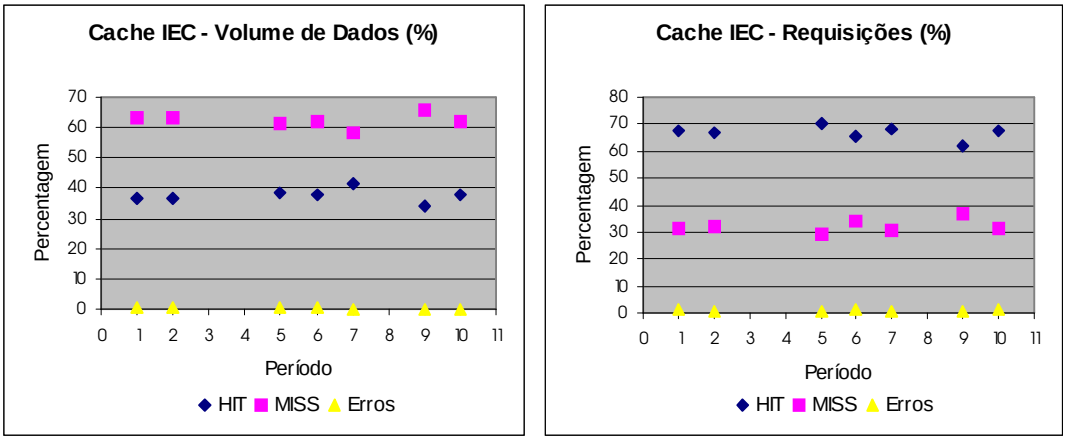


É interessante observar a inversão na taxa de acerto em número de requisições ocorridas neste servidor de *web-caching* e a estabilidade na taxa de acerto em volume de dados. Talvez esta inversão de uma taxa e a estabilidade de outra esteja associada a área em disco reservada para armazenagem dos objetos recuperados. Este é assunto que merece estudos futuros.

4.2.5.3 Resultados Cache IEC

Mantivemos os arquivos de eventos do servidor Cache IEC entre os dias 23 de setembro e 2 de dezembro de 2001. Os resultados obtidos entre o dia 23/09 e 11/11 estiveram sobre a Estrutura 1 apresentada na FIG. 4.3 e seus resultados médios semanais são apresentados do Período 1 a 7 na FIG. 4.7. Entre os dias 11/11 e 04/12 os servidores estavam dispostos de acordo com a Estrutura 2 apresentada na FIG. 4.4 e seus resultados médios semanais são apresentados nos Períodos 8 a 10 na FIG. 4.7.

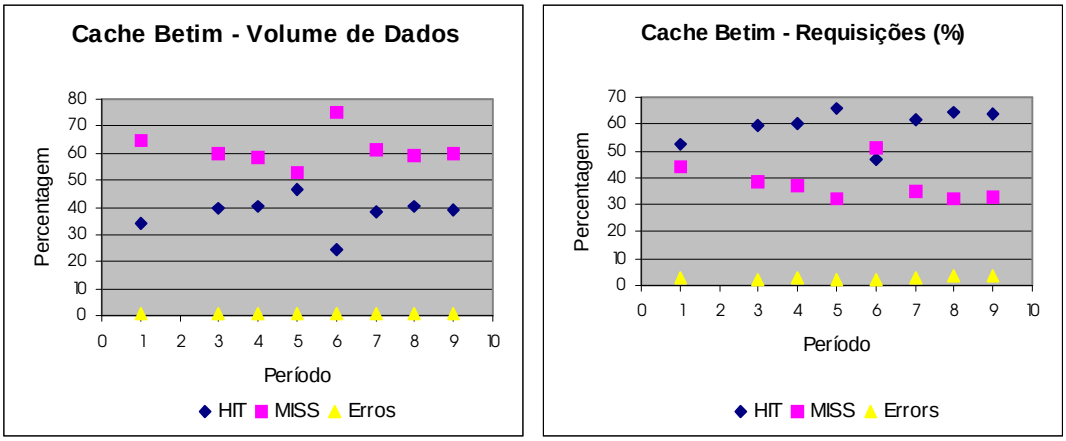
FIGURA 4.7 - Distribuição dos resultados do Servidor Cache IEC.



4.2.5.4 Resultados Cache Betim

Mantivemos os arquivos de eventos do servidor Cache Betim entre os dias 30 de setembro e 2 de dezembro de 2001. Os resultados obtidos entre o dia 30/09 e 11/11 estiveram sobre a Estrutura 1 apresentada na FIG. 4.3 e seus resultados médios semanais são apresentados do Período 1 a 6 na FIG. 4.8. Entre os dias 11/11 e 04/12 os servidores estavam dispostos de acordo com a Estrutura 2 apresentada na FIG. 4.4 e seus resultados médios semanais são apresentados do Período 7 a 9 na FIG. 4.8.

FIGURA 4.8 - Distribuição dos resultados do Servidor Cache Betim.

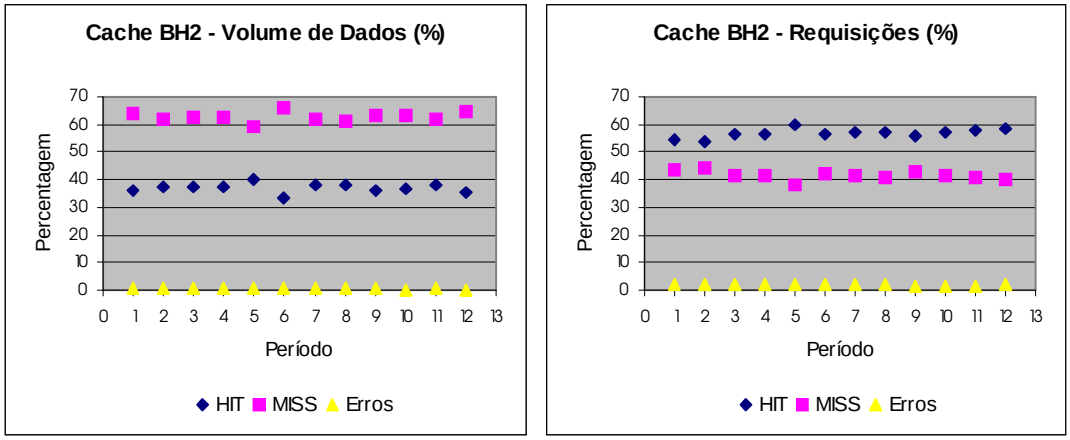


Durante a semana de 04 a 11 de novembro, período 6, a unidade de Betim promoveu a semana Linux, o que provocou grande variação em relação ao comportamento dos usuários e conseqüente reflexo na média calculada, justificando os valores apresentados na FIG. 4.8.

4.2.5.5 Resultados Cache BH2

Mantivemos os arquivos de eventos do servidor Cache BH2 entre os dias 09 de setembro e 2 de dezembro de 2001. Os resultados obtidos entre o dia 09/09 e 11/11 estiveram sobre a Estrutura 1 apresentada na FIG. 4.3 e seus resultados médios semanais são apresentados do Período 1 a 9 na FIG. 4.9. Entre os dias 11/11 e 04/12 os servidores estavam dispostos de acordo com a Estrutura 2 apresentada na FIG. 4.4 e seus resultados médios semanais são apresentados do Período 10 a 12 na FIG. 4.9.

FIGURA 4.9 - Distribuição dos resultados do Servidor Cache BH2.



4.2.5.6 Análise da Coleta de Dados

Analisando de forma geral as FIG. de 4.5 a 4.9, podemos verificar que o comportamento de cada um dos servidores é bastante homogêneo. Não existem grandes variações nos resultados coletados de um período para outro, exceto no caso já mencionado de Betim.

Os acertos em número de requisições atingem patamares elevados, em certas unidades até 70%, enquanto que os acertos em volumes de dados apresentam valores menores, mesmo assim expressivos, indicando que os acertos ocorreram com maior freqüência em arquivos pequenos e médios.

A média de volume de dados de acerto acima dos 30% permite afirmar que os servidores de *web-caching* assumiram bem as suas funções, reduziram significativamente o acesso às origens dos objetos requisitados e atenderam os usuários mais rapidamente, ou seja, diminuíram o tempo médio de reposta das transações.

A TAB. 4.1, apresentada a seguir, mostra o volume total e médio diário de requisições e volume de dados em *Mbytes* coletados durante todo o período de monitoração dos servidores *web-caching* da rede acadêmica da PUC, tanto na disposição da Estrutura 1 (FIG. 4.3) como na Estrutura 2 (FIG. 4.4). Os valores médios diários são apresentados em função do total coletado e a quantidade de dias em que os eventos foram registrados. Por exemplo, o Cache DCC disposto de acordo com a Estrutura 1 recebeu um total de 14.714.614 requisições durante os 56 dias acumulados nos arquivos de eventos coletados. A variação na quantidade de dias de cada servidor em cada estrutura é em razão da disponibilidade dos arquivos de eventos que foram coletados. Na Estrutura 1 os servidores Cache DCC e BH2 tinham mais arquivos de eventos coletados e na Estrutura 2 apenas o Cache IEC teve um dos arquivos de eventos perdido.

TABELA 4.1 - Resultados Coletados da Estrutura 1 e 2.

	Servidor	Dias	Total de Requisições	Média diária em requisições	Quantidade em MBytes	Média diária em MBytes
Estrutura 1	PUC	31	4.497.061	145.066	15.935	514
	DCC	56	14.714.614	262.761	63.856	1140
	IEC	35	1.679.105	47.974	4.354	124
	Betim	35	2.438.001	69.657	8.554	244
	BH2	62	3.317.454	53.507	12.085	194
Estrutura 2	PUC	23	5.551.765	241.381	20.405	887
	DCC	21	5.120.628	243.839	20.656	983
	IEC	14	783.607	55.971	2.119	151
	Betim	21	3.017.953	143.712	9.798	466
	BH2	21	1.245.053	59.288	4.302	204

A FIG. 4.10 apresenta o total de requisições e o volume de dados recolhidos durante a etapa de coleta de dados para cada um dos servidores

acompanhados. A FIG. 4.11 apresenta os mesmos parâmetros das figuras anteriores, porém em função da média diária. Todos estes valores estão disponíveis na TAB. 4.1.

Esta informação de média diária atendida pelos servidores *web-caching* procura eliminar problemas com períodos diferentes de coleta de dados para cada uma das estruturas e pode ser usada para estabelecer proporções entre quantidade de área em disco, que deve ficar disponível para armazenamento nos servidores, e total de tráfego gerado pelas requisições dos usuários.

FIGURA 4.10 - Distribuição total dos resultados coletados da Estrutura 1 e 2.

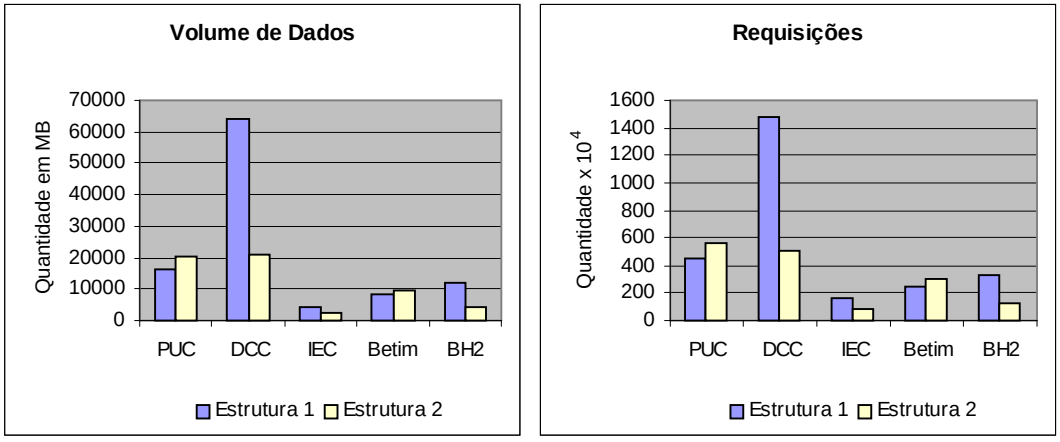
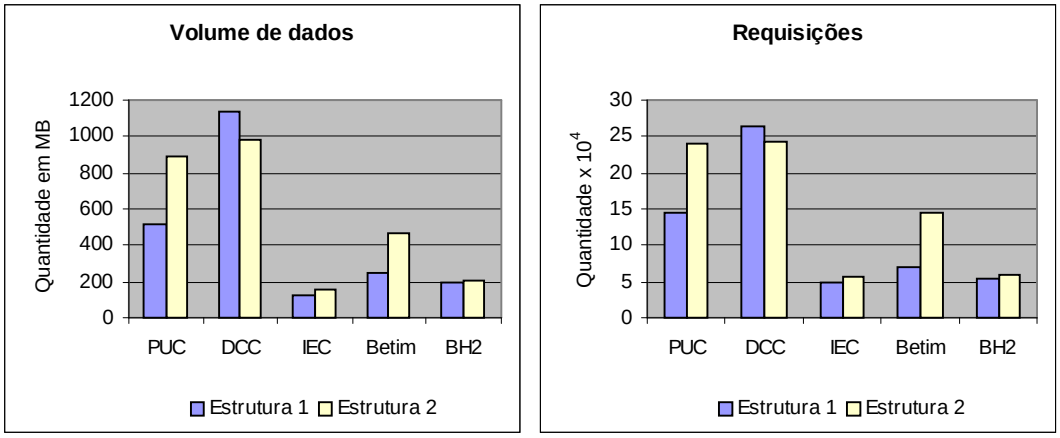


FIGURA 4.11 - Distribuição de média diária dos resultados coletados da Estrutura 1 e 2.

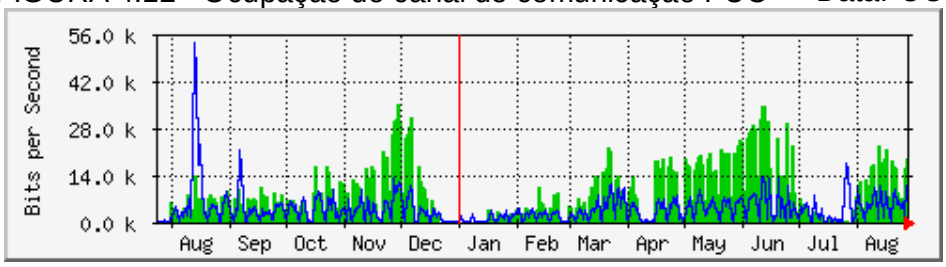


Analisando a FIG. 4.11 podemos verificar algumas tendências. O Cache PUC sofre um acréscimo em torno de 66% no número de requisições atendidas, tendo em vista que na Estrutura 2 são atendidas também requisições de outros servidores, atuando com pai e irmão. Além disso, o fator sazonalidade também estará influenciando (ver explicações do Cache IEC e BH2).

O Cache DCC tem um decréscimo em torno de 7%, devido ao final de semestre letivo quando alguns laboratórios ficam sem Internet, dedicados para programação e edição de documentos, conseqüentemente reduzindo o número de estações atendidas (sazonalidade).

Os Servidores Cache IEC e BH2 têm um maior volume de requisições, devido a elevação de consultas ao Sistema de Gestão Acadêmica (SGA) que divulga notas das disciplinas de todos os cursos. A FIG. 4.12 mostra a média diária dos últimos 13 meses da ocupação do *link* PUC Coração Eucarístico e DataPUC. As colunas cheias indicam tráfego do DataPUC para PUC e a linha contínua indica tráfego da PUC para DataPUC. No final de novembro e meados de junho o consumo de Internet sofre um considerável acréscimo no *link* de comunicação de dados dedicado a consultas do SGA, como pode ser comprovado pelas linhas cheias da figura a seguir.

FIGURA 4.12 - Ocupação do canal de comunicação PUC ⇔ DataPUC.



O Cache Betim, além de ter acréscimo na quantidade de requisições devido a sazonalidade como todos os outros servidores, sofreu grande influência da semana Linux e de outros fatores que não conseguimos identificar.

A TAB. 4.2 mostra os valores médios coletados do status das requisições em cada servidor de *web-caching*. O status atribuído a uma requisição pode ser: HIT – em caso de acerto na base web-*cache*; MISS – em caso de erro na base web-*cache*; Error – quando acontece algum erro na requisição em si.

Os valores obtidos para HIT (acertos) na TAB. 4.2 consideram apenas as requisições atendidas localmente, não fazendo parte da contagem as requisições atendidas por servidores pai(s) e/ou irmão(s). Desta forma é possível verificar o comportamento individual de cada servidor.

Os valores apresentados na Estrutura 1 e 2 são calculados em função da média ponderada dos resultados coletados em cada uma dos períodos onde os servidores de *web-caching* foram monitorados. Estes valores são apresentados em gráficos nas FIG. 4.13 e 4.14.

Também é apresentada na TAB. 4.2 a variação observada nos servidores *web-caching* de uma estrutura para outra em relação aos dois parâmetros monitorados.

TABELA 4.2 - Resultados Coletados.

		Estrutura 1		**Estrutura 2**		**Variação**	
Cache	**Status**	**RR**	**RKB**	**RR**	**RKB**	**RR**	**RKB**
PUC	**HIT**	54,9	28,84	52,24	27,68	2,66	1,16
	MISS	43,28	70,63	45,95	71,82	-2,67	-1,19
	ERROR	1,83	0,53	1,81	0,51	0,02	0,02
DCC	**HIT**	50,77	31,46	46,31	30,85	4,46	0,61
	MISS	46,36	67,54	50,13	67,86	-3,77	-0,32
	ERROR	2,86	0,99	3,56	1,28	-0,7	-0,29
IEC	**HIT**	68,16	38,22	65,35	36,2	2,81	2,02
	MISS	30,85	61,41	33,57	63,62	-2,72	-2,21
	ERROR	0,99	0,37	1,07	0,18	-0,08	0,19
Betim	**HIT**	58,08	38,26	63,27	39,16	-5,19	-0,9
	MISS	39,48	61,06	33,42	60,12	6,06	0,94
	ERROR	2,44	0,69	3,3	0,72	-0,86	-0,03
BH2	**HIT**	55,97	36,69	57,77	36,32	-1,8	0,37
	MISS	42,15	62,79	40,62	63,37	1,53	-0,58
	ERROR	1,89	0,52	1,61	0,32	0,28	0,2

RR – percentagem de requisições; **RKB** – percentagem em volume de dados.

Analisando a TAB. 4.2, em relação à variação de volume de dados (RKB), podemos observar alguns comportamentos interessantes. O Cache PUC tem variação pequena, justificada pela elevação das requisições atendidas e maior heterogeneidade dos usuários, pois na Estrutura 2 o Cache PUC também atende indiretamente usuários de outros *campi*. Na verdade, aguardávamos uma variação maior do que a observada. Os servidores Cache DCC, Betim e BH2 sofrem pouca variação, confirmando expectativas.

Em relação ao parâmetro de percentagem de requisições (RR), a variação coletada sofre oscilações consideráveis. Talvez esta característica esteja relacionada à área em disco reservada para armazenar objetos acessados, como observado na etapa de coleta de dados no caso do Cache DCC (item 4.2.5.2). Tal comportamento poderia ser avaliado em trabalhos futuros.

O Cache IEC sofreu maior variação, justificado pela periodicidade dos cursos desta unidade e pela menor quantidade de arquivos coletados. Este último motivo deixa a análise mais sensível a possíveis eventos extraordinários promovidos nesta unidade.

As FIG. 4.13 e 4.14, intituladas “Volume de Dados”, apresentam índices médios (em percentagem) do total de *bytes* atendidos nos servidores de *web-caching*. Valores resultantes do total de *HIT* (acertos), *MISS* (falhas) e Erros, observados durante o levantamento de dados da Estrutura 1 e 2 respectivamente (TAB. 4.2).

As FIG. 4.13 e 4.14, intituladas “Requisições”, apresentam os valores médios (em percentagem) da quantidade total de requisições atendidas nos servidores de *Web-caching*. Valores resultantes do total de *HIT* (acertos), *MISS* (falhas) e Erros, observados durante o levantamento de dados da Estrutura 1 e 2, respectivamente (TAB. 4.2).

FIGURA 4.13 - Distribuição resultados coletados da Estrutura 1.

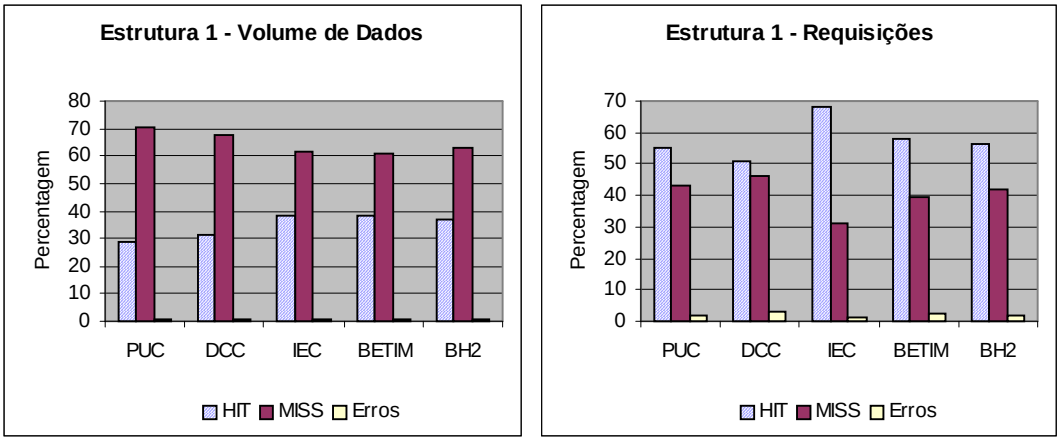
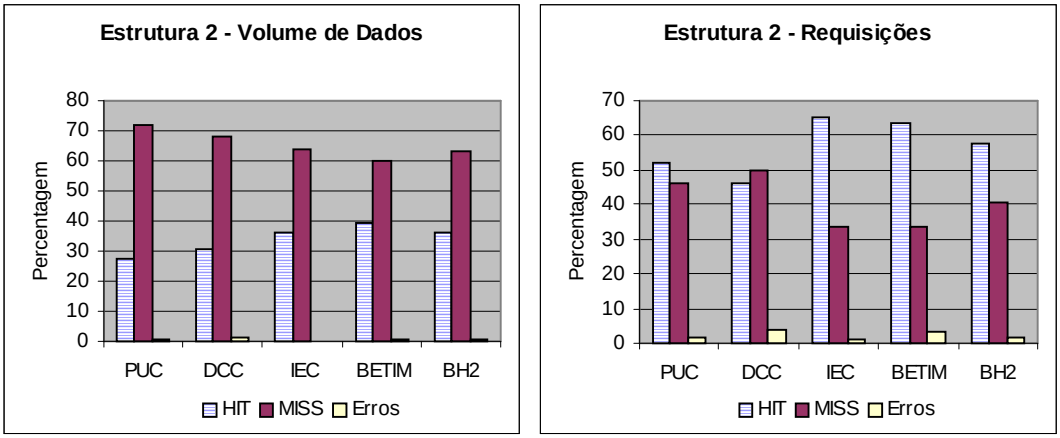


FIGURA 4.14 - Distribuição resultados coletados da Estrutura 2.



Analisando os gráficos e como já foi mencionado, em nosso estudo de caso, os acertos foram mais comuns em arquivos pequenos e médios, provocando uma elevada taxa de acerto em número de requisições e um resultado não tão bom assim na quantidade de *bytes* reaproveitados.

Ocorrendo um *miss* na Estrutura 2 o servidor w*eb-caching* tem três alternativas para recuperação do conteúdo desejado pelo usuário: direto da origem, hit no servidor pai ou irmão, ou recuperado pelo servidor pai. A TAB. 4.3 mostra a distribuição do comportamento de cada um dos servidores em função dessas características. Vale observar que um *hit* no Cache DCC, de uma requisição

enviada aos Cache IEC, Betim e BH2 conta com “Hit no irmão” do Cache PUC e como “Atendido pelo pai” nos servidores Cache IEC, Betim, BH2.

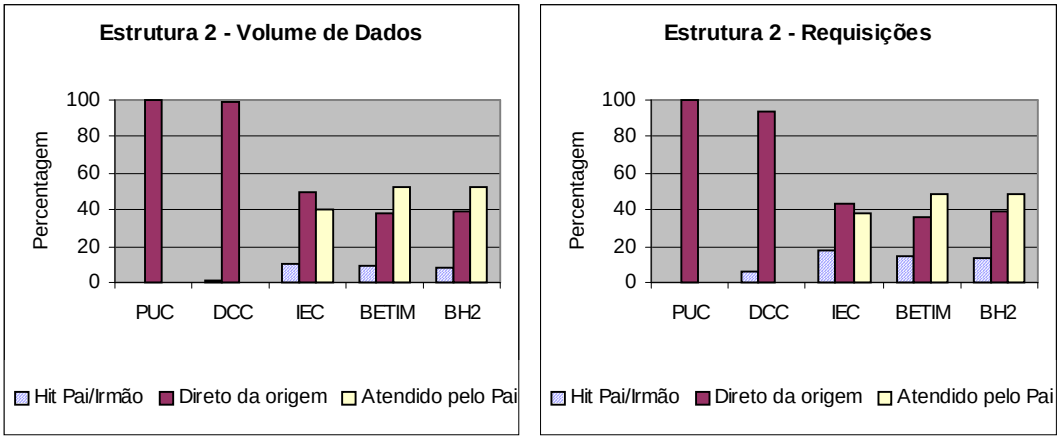
TABELA 4.3 - Resultados Coletados da Estrutura 2.

Servidor	Origens Consultadas	**RR**	**RKB**
PUC	Direto da origem	99,67	99,68
	Hit no irmão	0,33	0,32
	Atendido pelo pai	0	0
DCC	Direto da origem	94,02	99,11
	Hit no irmão	5,98	0,89
	Atendido pelo pai	0	0
IEC	Direto da origem	43,76	48,98
	Hit no pai	17,92	10,71
	Atendido pelo pai	38,32	40,32
Betim	Direto da origem	36,41	38,08
	Hit no pai	14,84	8,9
	Atendido pelo pai	48,76	53,02
BH2	Direto da origem	38,72	39,15
	Hit no pai	13,31	7,79
	Atendido pelo pai	47,97	53,05

RR – percentagem de requisições; **RKB** – percentagem em volume de dados.

Na FIG. 4.15 são apresentadas as distribuições médias, em percentagem, da quantidade de requisições e de volume de dados de onde cada um dos servidores de *Web-caching* recuperaram o objeto não armazenado localmente, dispostos de acordo com Estrutura 2.

FIGURA 4.15 - Distribuição resultados coletados da Estrutura 2, em função da forma como os servidores recuperaram objetos que causaram MISS no *web-caching* local.



No Cache PUC quase que a totalidade de falhas gerou um acesso ao servidor que hospedava o objeto na Internet. A colaboração de 'irmandade' estabelecida com o servidor Cache DCC não gerou bom aproveitamento. Podemos justificar como principal causa deste fato à sobrecarga do servidor Cache DCC que causou *timeout* nas consultas do Cache PUC (FIG. 4.15).

O servidor Cache DCC obteve maior aproveitamento da base mantida no servidor Cache PUC, reduzindo um pouco a necessidade de acesso ao servidor web origem do objeto, isto pode estar associado a maior capacidade de armazenamento e processamento do Cache PUC em relação ao Cache DCC. Como trabalho futuro podemos sugerir o estudo da influência de *timeout* na cooperação entre os servidores de uma estrutura hierárquica de *web-caching*.

Os servidores Cache IEC, Betim e BH2 tiveram um aproveitamento bem maior devido a colaboração estabelecida com o Cache PUC, o que já justificaria a implantação da Estrutura 2 ao invés da Estrutura 1.

Os servidores Cache BH2 e Betim se comportaram de forma muito similar. O Cache IEC, por sua vez, já precisou ir mais à origem do objeto que os outros dois servidores *web-caching* filhos. Isto é justificado pela velocidade do canal de comunicação dessas unidades ao *campus* principal. O canal do IEC é de apenas 64 Kbps o que retardava a resposta do servidor Cache PUC, ultrapassando o tempo de vida da requisição, ativando função de desvio para recuperação da informação direta de sua origem. Observamos que nos três servidores *cache* filhos o tempo de vida da requisição ao servidor pai estava abaixo do que realmente poderia ficar. O indicado é que a quantidade de acesso direto à origem seja o menor possível, aumentando a dependência ao servidor pai que conseqüentemente ficará com uma base de dados mais completa e atual.

No geral a qualidade dos dados coletados é muito boa. A sazonalidade e outras fontes de variações foram observadas, mas nada a ponto de invalidar os dados obtidos, ou melhor, serviram para demonstrar o comportamento real do ambiente.

A maior dificuldade nesta etapa é ter acesso aos arquivos de eventos dos servidores *web-caching*. Primeiro pelo fato do tamanho dos mesmos e a dificuldade de manusear arquivos com centenas de Mbytes, depois devido a recuperação desses arquivos que normalmente estão em unidades remotas com canais de comunicação de baixo desempenho, exigindo muito tempo para transmitir o arquivo de um lugar para outro.

Os objetivos da etapa dentro do método proposto foram alcançados. Os dados necessários para projetar o modelo de carga de trabalho e de desempenho são conhecidos. Dessa forma podemos dar continuidade ao método e considerar a etapa validada.

4.2.6 Modelo de Carga de Trabalho (M1)

A complexidade da modelagem da carga de trabalho foi maior do que esperávamos ao selecionar o estudo de caso. Diversas variáveis afetam o comportamento dos experimentos e a simulação de todos estes aspectos elevaria consideravelmente a quantidade de alternativas a serem validadas e conseqüentemente o tempo de experimentação. Sendo assim, resolvemos adotar um modelo de carga de trabalho mais simples em relação ao ambiente real e não trabalhar alguns parâmetros que podem influenciar no comportamento da estrutura.

Além disso, muitas simplificações foram necessárias devido a limitação apresentada pela ferramenta de simulação escolhida durante a etapa de seleção de aplicativos auxiliares (E5). Estas limitações e as conseqüentes

simplificações realizadas serão apresentadas durante a geração do modelo de carga de trabalho e na etapa de validação e calibração desse modelo.

Considerações importantes estabelecidas durante a modelagem da carga de trabalho a ser experimentada em etapas futuras:

- As limitações de tempo de resposta causadas pelos canais de comunicação de dados entre as diferentes unidades da PUC não foram consideradas, isto pode implicar em uma maior colaboração entre os servidores que não seriam observadas em um ambiente real;
- Não causaremos *stress* nas áreas de *cache*, ou seja, não nos preocuparemos com substituição de conteúdo quando o disco atingir seu limite. Os experimentos não consumirão toda a área reservada para armazenagem de dados. Esta simplificação não deverá causar grande impacto nas experimentações em função dos parâmetros em análise. Considere que seja planejado *stress* na área em disco, para alcançarmos uma taxa de acerto x, teríamos que aumentar a taxa de repetição de requisições geradas pelos processos clientes do ambiente simulado, esta elevação na taxa de repetição compensaria as falhas causadas por requisições que teriam sido removidas da área em disco por falta de espaço para armazenar novos objetos, induzindo a resultados semelhantes aos experimentos que não causarem *stress*;
- A freqüência de requisições por segundo disparadas a cada servidor será constante. Esta convenção é adotada em função das características do aplicativo usado para gerar as requisições. Este assunto nos leva a discussão de capacidade de atendimento dos servidores e pode se tornar um assunto para trabalhos futuros. A dissertação de FONSECA [31] aborda este assunto.
- As distribuições e as características dos objetos que farão parte do modelo de carga de trabalho não seguem comportamento verificado durante a etapa de coleta de dados, serão utilizados índices padrões de distribuições e características dos objetos da ferramenta escolhida para simulação. Maiores detalhes serão apresentados posteriormente.

O modelo de carga de trabalho se caracteriza pelo estabelecimento de valores das variáveis que controlam o comportamento dos processos que simulam os clientes e os servidores *web* de conteúdo que formam a estrutura apresentada na FIG 3.2 (Ambiente Simulado).

A primeira etapa no processo de modelagem da carga de trabalho foi extrair dos arquivos de eventos coletados durante a etapa E5 da Estrutura 1 (FIG. 4.3) a taxa média ponderada de acerto em quantidade de requisições e taxa média ponderada de acerto em volume de dados para cada servidor sob estudo. Estes valores são apresentados na TAB. 4.4 e com os mesmos será possível iniciar à definição de distribuições e características dos objetos que na realidade devem compor nosso modelo de carga de trabalho sintético.

TABELA 4.4 - Resultados Coletados da Estrutura 1.

Servidor	REQ	HREQ	HRR	KB	HKB	HRKB
PUC	4497061	2468886	54,90	16317440	4705950	28,84
DCC	14714643	7470624	50,77	65388544	20571236	31,46
IEC	1679105	1144478	68,16	4459433	1614315	36,20
Betim	2438001	1415991	58,08	8759718	3430306	39,16
BH2	3317454	1856779	55,97	12375040	4494615	36,32

REQ – total de requisições atendidas; **KB** - volume total de dados atendido; **HREQ** – total de requisições que geraram hit; **HRR** – percentagem de hit em requisições; **HKB** – volume total de dados que gerou hit; **HRKB** – percentagem de hit em volume de dados.

Limitamos as requisições a três tipos distintos de objetos: imagem, html e arquivo de *download*. Cada um com três características definidas: tamanho fixo em KB, *cachable* (taxa de armazenagem, ou seja, indica a percentagem de objetos que podem ser armazenados), recorrência (taxa de repetição na consulta de cada objeto, ou ainda, distribuição dos objetos). Esta modelagem segue sugestões de modelos de cargas sintéticos que acompanham o aplicativo que realiza a simulação escolhida na etapa E4. Outros aplicativos descritos no Capítulo 2 utilizam sugestões de cargas similares. Novamente, devemos mencionar que estes objetos não são representativos em relação aos

objetos reais do ambiente sob estudo selecionado e sim representam objetos sintéticos padrões utilizados por ferramentas de simulação.

Apresentando quantitativamente as características desses objetos, temos:

- Imagem:
  - Tamanho: 4.5 Kbytes;
  - *Cachable*: 80%, ou ainda, 20% dos objetos imagem gerados não podem ser armazenados;
  - Recorrência: 75%;

- HTML:
  - Tamanho: 8.5 Kbytes;
  - *Cachable*: 90%;
  - Recorrência: 23%;

- Arquivos de *download*:
  - Tamanho: 225 Kbytes;
  - *Cachable* médio: 20%;
  - Recorrência: 2%.

Os índices descritos acima seguem padrões estabelecidos pela organização que desenvolveu o “Web-Polygraph” [79] e pela caracterização de carga descrita no Capítulo 2, exceto o atributo *cachable* dos objetos arquivos de *download*. Este atributo é calculado em função dos atributos de outros objetos e da taxa de acerto em volume de dados que se pretendia atingir. Como as taxas de acertos em volume de dados são diferentes para cada servidor *web-caching* o tamanho e *cachable* dos objetos arquivos de *download* são diferentes para cada carga de trabalho de cada servidor. Em média este valor permanecia em 20%.

Outro índice que deve variar de uma carga de trabalho para outra é o índice de repetitividade dos objetos que os processos clientes do aplicativo de simulação

devem produzir. Este índice é calculado em função das características de todos os objetos e da taxa de acerto média em quantidade de requisições desejada para cada par de processo cliente e servidor web que representa um servidor de *web-caching* real apresentada na TAB. 4.4.

A escolha de três objetos tem por objetivo possibilitar carga de objetos de diferentes tamanhos, aproximando o modelo de carga sintética à carga real, além de permitir ajustes dos parâmetros com o intuito de atingir os valores apresentados na TAB. 4.4.

A duração e a quantidade de requisições disparadas por segundo a cada um dos servidores *web-caching* devem ser obtidos durante a etapa de validação e calibração do modelo de carga de trabalho, quando acompanhamos o comportamento das estruturas simuladas no transcorrer do tempo. Considerando estruturas com hierarquia, temos que os servidores pais e irmãos terão mais requisições, tendo em vista o fato deles receberem requisições de outros servidores e não só do processo cliente.

O processo cliente do aplicativo de simulação escolhido deve ser calibrado para obter a taxa de acerto em número de requisições dentro do observado (TAB. 4.4); gerar colaboração entre servidores semelhante aos obtidos na Estrutura 2 (FIG. 4.4, TAB. 4.3) avaliada; determinar duração do experimento para estabilização do comportamento dos servidores; determinar número máximo de requisições por segundo que não comprometa capacidade de processamento dos servidores, permitindo simulações com tempo menor.

As características que geram colaboração entre servidores semelhante aos obtidos na Estrutura 2 (FIG. 44, TAB 4.3) ainda não foram modelados e serão discutidos durante a etapa de validação e calibração do modelo de carga (E6), pois na prática a etapa de modelagem da carga de trabalho (M1) acontece em paralelo a etapa de validação e calibração (E6).

4.2.7 Validação e Calibração do Modelo de Carga (E6)

Neste momento do método devíamos definir a estrutura de rede e hardware que iríamos utilizar. Como previíamos que as experimentações seriam de longa duração, não poderíamos contar com a estrutura que se encontrava em produção, muito menos contar com vários servidores. A solução encontrada foi selecionada em função do número de computadores que teríamos a disposição, neste caso dois.

A etapa de validação e calibração do modelo de carga de trabalho foi dividida em duas sub-etapas. Na primeira sub-etapa levamos em consideração os valores recolhidos com os servidores de *web-caching* dispostos de acordo com a Estrutura 1 (FIG. 4.3). Esta validação tinha por objetivo verificar o comportamento da ferramenta selecionada para simular os processos clientes e servidores. A outra sub-etapa deve levar em consideração a necessidade de relação entre os processos da ferramenta de simulação para que seja possível promover acertos dentro da estrutura hierárquica e não somente *no web-caching* local de cada um dos servidores.

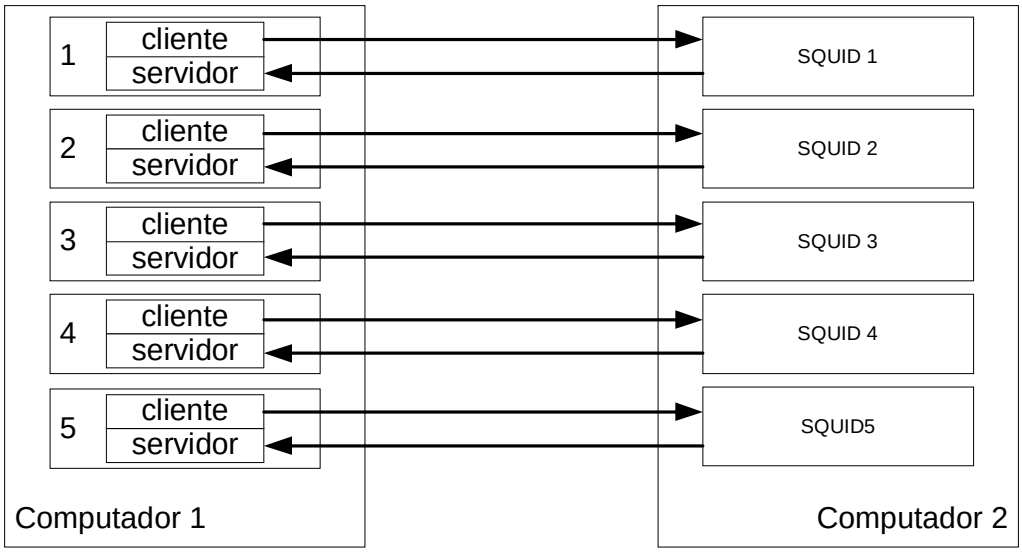
Durante esta primeira sub-etapa conseguimos identificar quantas requisições seriam necessárias para estabilizar os resultados, qual a variação esperada depois de modificadas características dos objetos e qual seria o comportamento geral dos processos. Estas características são muito importantes para a etapa como um todo.

Nesta sub-etapa as simulações foram feitas com cinco pares de processos cliente-servidor. Os pares de processos se distinguiam entre si na repetitividade gerada pelos processos clientes e na propriedade de "*cachable"* dos objetos "arquivos de *download*" criados pelo processo servidor. A manipulação destes dois parâmetros são suficientes para aproximar o comportamento do servidor *web-caching* no ambiente simulado ao observado no sistema real.

Fisicamente colocamos um dos computadores dedicados à simulação com dez processos rodando, um em cada porta TCP/IP, cinco processos clientes e cinco servidores. No outro servidor colocamos cinco processos do SQUID executando, cada um simulando um servidor de *web-caching* distinto e cada um trabalhando em uma porta TCP/IP distinta. Esta configuração nos permitiu com pouco *hardware* simular vários servidores, em compensação não podíamos solicitar muitas requisições simultaneamente, pois o servidor atingia sua capacidade de processamento, refugando algumas requisições.

Observamos que acima de 100 requisições por segundo, no total, o computador com os processos SQUID atingia seu limite depois de alguns minutos. Portanto definimos que cada processo cliente deveria requisitar 10 objetos por segundo ao seu respectivo servidor SQUID. A figura a seguir mostra a lógica da estrutura experimental em laboratório montada para execução das simulações nesta primeira sub-etapa.

FIGURA 4.16 - Estrutura lógica da simulação – Primeiro Modelo.



Cada processo cliente interage com seu respectivo processo SQUID solicitando um objeto que deve ser recuperado de um processo servidor específico, por isso identificamos na figura cinco grupos de processos. Os

processos clientes disparam as requisições simultaneamente a uma taxa de 10 requisições por segundo, tempo suficiente para que todos os processos SQUID processem e retornem o resultado, finalizando todas as transações antes do próximo conjunto de requisições.

A TAB. 4.5 apresenta os valores observados durante a coleta de dados da Estrutura 1 (ambiente real) e os valores obtidos em diversas simulações, onde variamos apenas a quantidade total de requisições disparadas a cada um dos servidores, mantendo os atributos de carga dos objetos inalterados.

TABELA 4.5 - Resultado Ambiente real e simulado

		PUC	DCC	IEC	Betim	BH2
Ambiente Real	**HRR**	54,90	50,77	68,16	58,08	55,97
	HRKB	28,84	31,46	36,20	39,16	36,32
15.000 req.	**HRR**	54,80	50,41	68,22	58,49	56,12
	HRKB	30,21	34,59	40,14	40,95	35,92
72.000 req.	**HRR**	54,55	50,62	68,62	57,95	55,93
	HRKB	30,11	32,68	36,43	39,08	35,89
150.000 req.	**HRR**	54,78	50,62	68,27	57,90	55,79
	HRKB	27,59	32,41	37,47	38,62	35,60
225.000 req.	**HRR**	54,69	49,65	68,05	57,50	55,74
	HRKB	28,80	32,16	37,01	39,73	35,64
360.000 req.	**HRR**	54,50	49,36	67,94	57,21	54,95
	HRKB	28,66	31,93	36,41	38,04	36,94

Durante a validação do modelo de carga de trabalho não observamos uma convergência nos resultados obtidos que justificasse a elaboração de simulações extremamente longas e com grande quantidade de requisições. Ao contrário, aumentando o número de requisições geradas, a área de disco destinada ao armazenamento dos dados atingia seu limite, provocando substituição de arquivos, elevando a distorção e dificultando ainda mais a definição dos parâmetros que formam o modelo de carga de trabalho. Não chegamos a simular estruturas que causassem stress na área em disco.

É importante observar que a falta de convergência dos resultados indica uma fragilidade no aplicativo selecionado para simular requisições ao sistema. A expectativa era que quanto maior o número de requisições menor seria a

distorção em relação aos valores observados na coleta de dados, e isto não se comprovou.

A TAB. 4.6 mostra a diferença em percentagem entre o valor desejado (ambiente real) que se queria simular e o valor obtido (ambiente simulado). Resolvemos adotar simulações com a menor quantidade de requisições em que os resultados simulados não extrapolassem 5% do valor desejado. Sendo assim, selecionamos simulações com 72.000 requisições. Tendo definido anteriormente que cada servidor pode receber 10 requisições por segundo, temos que o tempo total de cada experimentação deve ser de duas horas de duração.

TABELA 4.6 - Resultados do Ambiente real e simulado.

		PUC	**DCC**	**IEC**	**Betim**	**BH2**
15.000 req.	**HRR**	0,18	0,71	-0,09	-0,71	-0,27
	HRKB	-4,75	-9,95	-10,88	-4,57	1,10
72.000 req.	**HRR**	0,64	0,30	-0,67	0,22	0,07
	HRKB	-4,40	-3,88	-0,64	0,20	1,18
150.000 req.	**HRR**	0,22	0,30	-0,16	0,31	0,32
	HRKB	4,3	-3,02	-3,51	1,38	1,98
225.000 req.	**HRR**	0,38	2,21	0,16	1,0	0,41
	HRKB	0,14	-2,23	-2,24	-1,46	1,87
360.000 req.	**HRR**	0,73	2,78	0,32	1,50	1,82
	HRKB	0,62	-1,49	-0,58	2,86	-1,71

O modelo de carga de trabalho estaria satisfatório caso não houvesse necessidade de garantir uma interação entre as consultas geradas entre os diferentes processos clientes. Os pares de processos cliente-servidor não tinham relacionamento entre si e qualquer hierarquia estabelecida não apresentaria um comportamento semelhante ao ambiente real. Isto porque uma falha ocorrida em um servidor de *web-caching* da hierarquia geraria falha em todos os outros servidores da estrutura.

Tendo em vista esta restrição do primeiro modelo de carga de trabalho partimos para validação de um “segundo modelo”, dando início a segunda sub-etapa da etapa E6, este segundo modelo é mais adequado para nosso estudo

de caso. Baseamos o novo modelo em um único processo servidor e um único processo cliente, ambos rodando em um único computador em portas TCP/IP distintas como na primeira sub-etapa. Este processo cliente é dividido em cinco robôs de geração de requisições, permitindo consultas repetidas entre os robôs e com cada robô gerando consultas para um servidor de w*eb-caching* específico.

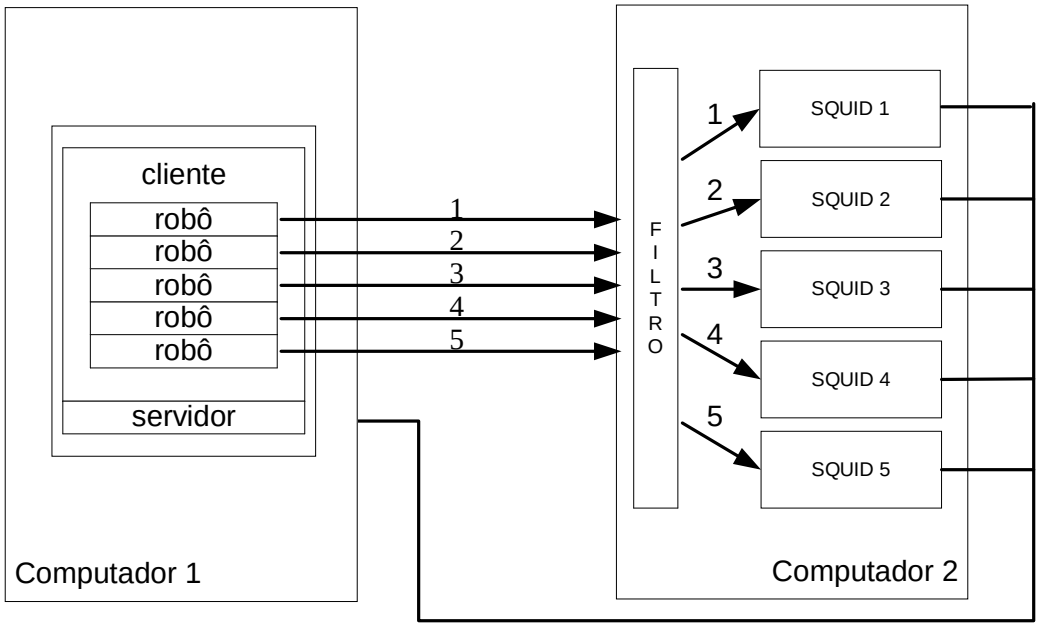
O aplicativo “Web-Polygraph” não tem como característica permitir a identificação de qual robô vai interagir com qual processo servidor de *web-caching*. Para resolver este problema usamos outra característica do aplicativo que permite atribuir a cada robô um endereço IP que será utilizado para identificar origem da requisição.

Conhecendo o IP de origem de cada robô do processo cliente foi possível criar um filtro no computador 2 que associa um robô a um processo SQUID, simulando uma estrutura com vários servidores *web-caching*, vários usuários e vários servidores de conteúdo.

A FIG. 4.17 apresenta a estrutura usada para simular todas as experimentações realizadas a partir desta sub-etapa em nosso estudo de caso.

Os processos SQUID não necessariamente recuperam os objetos no processo servidor do Computador 1, isto depende da estrutura hierárquica que está sob simulação. Considere que o processo squid 1 representa um servidor *web-caching* filho do servidor *web-caching* representado pelo processo squid 2. Neste caso, antes do processo squid 1 requisitar o objeto ao processo servidor no Computador 1, existe uma consulta ao processo squid 2 que pode ou não gerar um acerto no web-*cache* do servidor pai. Em caso de resposta negativa do pai o processo squid 1 pode requisitar o objeto ao processo servidor ou mesmo ser atendido pelo servidor *web-caching* pai representado em nosso exemplo pelo processo squid 2.

FIGURA 4.17 - Estrutura lógica da simulação – Segundo Modelo



O principal problema neste segundo modelo estava em configurar diferentes pares de taxa de acerto em quantidade de requisições e taxa de acerto em volume de dados para cada servidor de *web-caching*, assim como coletado na etapa E5 e apresentada na TAB 4.4. A ferramenta não possibilita tal configuração, ou não conseguimos identificar como realizar esta configuração. Seria preciso descartar requisições geradas pelos robôs de tal maneira a aproximar os resultados do ambiente simulado ao ambiente real para cada um dos servidores *web-caching*.

Nossa opção foi trabalhar com taxas de acertos em quantidade de requisições e volume de dados iguais para todos os robôs, tornando nosso modelo não representativo em relação à carga real coletada, esta simplificação foi devida a uma limitação da ferramenta de simulação escolhida.

No entanto, devemos esclarecer que não nos desviamos de nosso objetivo principal com o estudo de caso que é comprovar a eficiência do método proposto, o que estamos comprometendo é a representatividade do modelo de

carga de trabalho e isto deve implicar em grandes distorções de resultados quantitativos nas próximas etapas. Como conseqüência direta dessas simplificações do modelo de carga de trabalho devemos verificar na etapa de validação em produção (E11) da estrutura escolhida na etapa E10 que a estrutura escolhida não apresenta o comportamento esperado sendo necessário retornar ao início do método indicado pela seqüência E11.R21.E1.

Tendo nos decidido a utilizar robôs gerando as mesmas taxas de acertos e quantidade de requisições (HRR) e volume de dados (HRKB), partimos para o cálculo desses valores únicos. Utilizamos os dados recolhidos durante o período no qual o conjunto de servidores *Web-caching* estava disposto de acordo com a Estrutura 1 (FIG. 4.3), já que esta estrutura não sofria influência de colaboração entre os servidores. Os valores médios ponderados foram calculados de acordo com as fórmulas a seguir:

$$\text{Média HRR} = \frac{\sum_{i=\text{Servidor}} REQ_i * HRR_i}{\sum_{i=\text{Servidor}} REQ_i}$$

$$\text{Média HRKB} = \frac{\sum_{i=\text{Servidor}} KB_i * HRKB_i}{\sum_{i=\text{Servidor}} KB_i}$$

Os valores utilizados nas fórmulas para o cálculo da taxa de acerto médio em quantidade de requisições (HRR) e taxa de acerto médio em volume de dados (HRKB) estão na TAB 4.4 e os resultados são HRR = 53,88% e HRKB = 32,45%.

Com a coleta de dados da Estrutura 2 (FIG. 4.4.) aproveitamos os resultados de colaboração obtidos dos servidores Cache IEC, Betim e BH2 com o servidor Cache PUC e extraímos a média já que tínhamos também a limitação da

ferramenta para aplicar diferentes índices. Este valor nos indicou qual deveria ser a recorrência entre os robôs para simular o ambiente real, ou seja, conseguimos determinar em quanto deveria ser o acerto na base do servidor Cache PUC em caso de falha nos servidores Cache IEC, Betim e BH2. Isto para validar o segundo modelo de carga de trabalho para a Estrutura 2.

Portanto, o modelo de carga de trabalho escolhido tem como características aproximar a percentagem de requisições atendidas (HRR) em cada um dos servidores de *web-caching* a 53,88%, a percentagem em *bytes* (HRKB) a 32,45% e a colaboração entre pai e filho a 14,5% em quantidade de requisições. Observe que novamente por limitação da ferramenta de simulação não tínhamos como estipular a taxa de acerto em volume de dados esperada na relação do servidor pai e filho.

Além de todas estas limitações apresentadas pela ferramenta de simulação escolhida, verificamos que as simulações também não convergiam para valores exatos. Podemos comprovar isto ao verificar os resultados do primeiro modelo de carga estabelecido. No caso do segundo modelo, em que existe uma relação mais estreita entre os robôs de geração, a distorção, em valores reais, é ainda maior do que a observada no primeiro modelo. Em alguns casos, sem explicação aparente, os resultados em percentagem de KB atendidos localmente divergiam muito do desejado, mesmo mantendo a carga de trabalho idêntica entre duas experimentações.

Devido a esta característica, criamos uma convenção muito importante na qual descartávamos toda experimentação onde o volume de dados atendidos localmente por um servidor *web-caching* filho extrapolava uma margem de ± 5% do valor desejado, em valores reais, esta margem de erro aceitável fica entre ± 1,62. Ou ainda, experimentações em que pelo menos um dos servidores *web-caching*, que só atendia requisições originadas do processo cliente, ficasse com os resultados de HRKB fora do intervalo 30,83% e 34,07% a experimentação era descartada.

Somente os resultados de taxa de acerto em volume de dados apresentavam tal instabilidade, os outros resultados permaneceram dentro do esperado. Esta instabilidade pode ter sido causada por alguma falha na implementação do aplicativo selecionado ou por uma interpretação errada por nossa parte. A realidade era que dois experimentos, executados em seqüência com todos os parâmetros de carga idênticos, produziam resultados que variavam em até 10% de uma simulação para outra.

Resumindo, temos o modelo de carga de trabalho caracterizado da seguinte forma. Um único processo servidor e um único processo cliente. O processo cliente composto por cinco robôs, cada um disparando requisições para um servidor de *web-caching* diferente. Os objetos gerados eram do tipo imagem, html e arquivo de *download*, com a distribuição de 75%, 23% e 2% respectivamente. O tamanho de cada objeto foi definido em 4.5 KB, 8.5KB e 225KB e o índice de *cachable* em 80%, 90% e 20%, respectivamente.

Cada robô apresentava uma recorrência nas consultas geradas de 80% e um interesse público de 39%, este interesse público foi determinado de forma empírica até atingirmos a taxa colaboração de 14,5% entre servidor pai e filho na simulação da Estrutura 2. O parâmetro de recorrência não indica diretamente qual será a taxa de acerto obtida, já que uma parte dos objetos não é armazenada na área de *web-caching* devido as suas características de *cachable*, e sim indica qual é a taxa de repetição que deve ser produzida. O parâmetro “interesse público” é que indica ao processo cliente qual deverá ser a taxa de repetição gerada entre os diversos robôs. Assim como a recorrência, este valor não indica exatamente a taxa de acerto que será obtida devido à característica dos objetos não serem completamente armazenáveis.

A TAB. 4.7 apresenta os resultados em percentagem de *hit* em requisições (HRR), *hit* em kilobytes (HRKB) e *hit* em requisições no servidor pai (HPIR) observados nos servidores Cache IEC, Betim e BH2 - servidores que atendem somente requisições geradas pelo processo cliente da ferramenta de

simulação. Estes resultados servem para validação do segundo modelo de carga para estrutura hierárquica dispostas de acordo com a Estrutura 2 (FIG. 4.4), aproximando os resultados à média ponderada calculada a partir da Estrutura 1 (FIG. 4.3). A última coluna indica se a experimentação seria ou não descartada durante a etapa de simulação. Os valores em negrito indicam resultados fora da margem aceitável, ou seja, acima de 5% de distorção.

TABELA 4.7 - Resultados de validação modelo de carga de trabalho.

	IEC			**BETIM**			**BH2**			**Status**
	HRR	**HRKB**	**HPIR**	**HRR**	**HRKB**	**HPIR**	**HRR**	**HRKB**	**HPIR**	
Experimento 01	53.77	33.83	14.52	53.74	33.65	14.56	53.84	**34.11**	14.58	Rejeitado
Experimento 02	53.39	33.61	14.18	53.80	**34.14**	14.55	53.42	32.18	14.51	Rejeitado
Experimento 03	53.56	**35.13**	14.51	53.59	**34.38**	14.93	53.97	**34.47**	14.66	Rejeitado
Experimento 04	54.58	**34.73**	14.94	54.32	**34.64**	15.07	54.76	**35.25**	14.92	Rejeitado
Experimento 05	54.09	33.19	14.59	54.17	33.92	15.05	54.02	33.13	14.64	Aceito
Experimento 06	53.52	**30.05**	14.52	53.87	**29.39**	14.61	53.97	31.16	14.91	Rejeitado
Experimento 07	53.11	**30.08**	14.50	53.88	**30.54**	14.80	53.65	**30.49**	14.90	Rejeitado
Experimento 08	53.06	32.45	14.43	53.14	32.56	14.46	53.55	32.43	14.47	Aceito
Experimento 09	53.88	**34.55**	15.02	53.72	33.61	14.79	53.65	32.50	14.46	Rejeitado
Experimento 10	53.67	31.94	14.61	53.85	32.24	14.75	53.41	32.79	14.52	Aceito
Experimento 11	53.45	31.22	14.62	53.69	32.70	14.57	53.65	33.06	14.74	Aceito
Experimento 12	53.75	32.02	14.45	53.87	32.76	14.28	53.44	31.67	14.42	Aceito
Experimento 13	53.30	33.27	14.46	53.70	31.99	15.09	53.33	31.75	14.52	Aceito
Experimento 14	53.73	31.77	14.98	53.24	31.09	14.45	54.00	31.98	14.87	Aceito
Experimento 15	54.03	33.10	14.86	53.51	33.07	14.63	53.67	33.95	14.71	Aceito

Observamos que variando o tamanho do objeto “arquivo de *download*” em 0,1 *Kbyte* de um experimento para outro, o comportamento da variável taxa de acerto em *kilobytes* (HRKB) era menor e conseqüentemente minimizávamos a perda das experimentações. Esta técnica foi adotada nas outras experimentações com o objetivo de reduzir perdas.

Esta etapa exigiu muitos cuidados e considerações importantes que influenciam todas as simulações que devem ser executadas na etapa de simulação (E9).

As simplificações merecem críticas: não foram consideradas as capacidades dos canais de comunicação de dados entre as unidades; a carga uniforme para todos os servidores de *web-caching* não representa a realidade; a capacidade física de cada servidor não foi modelada; as características dos objetos não foram comparadas aos obtidos na coleta de dados; a taxa de acerto nos servidores *web-caching* pais são constantes assim como a carga é uniforme em todos os servidores *web-caching*. Um modelo de carga mais representativo para análise de desempenho de servidores de *web-caching* reduziria a probabilidade de problemas na etapa de validação em produção (E11).

O aplicativo de simulação escolhido, também merece crítica: os resultados das simulações não convergiram; simulações com mesma carga de trabalho apresentavam resultados bastante diferentes; não é possível associar um processo robô a um processo servidor web-caching específico; não é possível associar taxas de acertos em quantidade de requisições e volumes de dados diferentes a cada robô.

Apesar do modelo não ser completamente representativo em relação ao ambiente real, o que deve ocasionar muitas distorções nas próximas etapas e impedir uma análise de desempenho correta para nosso estudo de caso, nosso objetivo principal com o estudo de caso está sendo alcançado, ou seja, estamos conseguindo validar as etapas e modelos e comprovar a eficiência do método proposto.

4.2.8 Modelo de Desempenho (M2)

O modelo de desempenho para o estudo de caso da Rede Acadêmica da PUC-Minas deve ser formulado levando-se em consideração o modelo de carga definido para as simulações, os recursos de *hardware* disponíveis e os recursos de *software*s utilizados. Estas três componentes devem permitir a formulação de um modelo que resulte em valores quantitativos que permitirão o

estabelecimento de comparações entre as estruturas hierárquicas de *web-caching* definidas na etapa E8.

Como o modelo de carga de trabalho especificado é comum para todas as estruturas propostas, a definição do modelo de desempenho fica simplificada. Na verdade, o modelo de carga de trabalho estabelece o comportamento de desempenho em cada servidor filho em sua base *web-cache* local. Neste caso, a incógnita a ser avaliada é a variação de comportamento dos servidores pais e irmãos e qual o reflexo no desempenho geral da estrutura.

Os recursos de *hardware* e *softwares* são os mesmos para todos os experimentos. Não aplicamos variações em área reservada para armazenamento ou capacidade de processamento dos servidores. As características dos canais de comunicações entre os servidores também foram abstraídas, ou seja, não estaremos simulando os efeitos causados por canais de comunicação com maior ou menor capacidade de transmissão.

Neste contexto, o modelo de desempenho pode ser caracterizado em função dos resultados das taxas de erros em número de requisições ou quantidades de *bytes*. Podemos considerar como estrutura mais eficiente aquela que exigir menor acesso aos servidores de origem da requisição solicitada.

Analiticamente, a avaliação será obtida da comparação dos somatórios de volume de dados obtidos diretamente da origem. Os resultados em volume de dados obtidos direto da origem para cada um dos servidores que participam na estrutura serão acumulados. Comparados estes somatórios, a estrutura com maior eficiência será aquela com menor valor de somatório.

Esta formulação implica em determinar qual estrutura exige menor consulta à Internet, conseqüentemente necessitando menos investimentos no recurso mais caro da estrutura - canal Internet - e reduzindo a latência de respostas às requisições geradas pelos usuários em média dos acessos.

Considerando que nosso modelo de desempenho baseia-se no volume de dados recuperados direto da origem e tendo em vista que já concluímos que a Estrutura 2 apresenta melhor desempenho em comparação à Estrutura 1 por recuperar um volume de dados menos da origem dos objetos, podemos considerar nosso modelo de desempenho validado.

4.3 Experimentação

4.3.1 Definição de Estruturas Hierárquicas para Simulação (E8)

Para o estudo de caso da PUC-Minas, resolvemos experimentar cinco estruturas hierárquicas diferentes, que teoricamente devem apresentar os melhores desempenhos. Esta escolha foi baseada nos argumentos apresentados no Capítulo 3. Outras opções de estruturas hierárquicas iriam considerar colaboração entre os servidores Cache IEC, Betim e BH2, porém a aplicação de uma destas estruturas em produção não se justifica pela limitação dos canais de comunicação entre as unidades.

Duas estruturas serviram para validar o modelo de carga de trabalho na etapa E6 e foram novamente simuladas na fase de experimentação, são elas: a Estrutura 1 (FIG. 4.3) e a Estrutura 2 (FIG. 4.4).

As outras três estruturas hierárquicas propostas serão denominadas Estrutura 3, 4 e 5. A Estrutura 3 assemelhasse a Estrutura 2 com a seguinte diferença: os servidores Cache IEC, Betim e BH2 além de serem filhos do servidor Cache PUC, também serão filhos do servidor Cache DCC. A relação de irmandade entre o Cache PUC e Cache DCC permanecerá idêntica. Esta estrutura foi escolhida pelo fato de possuir alta cooperação entre os servidores.

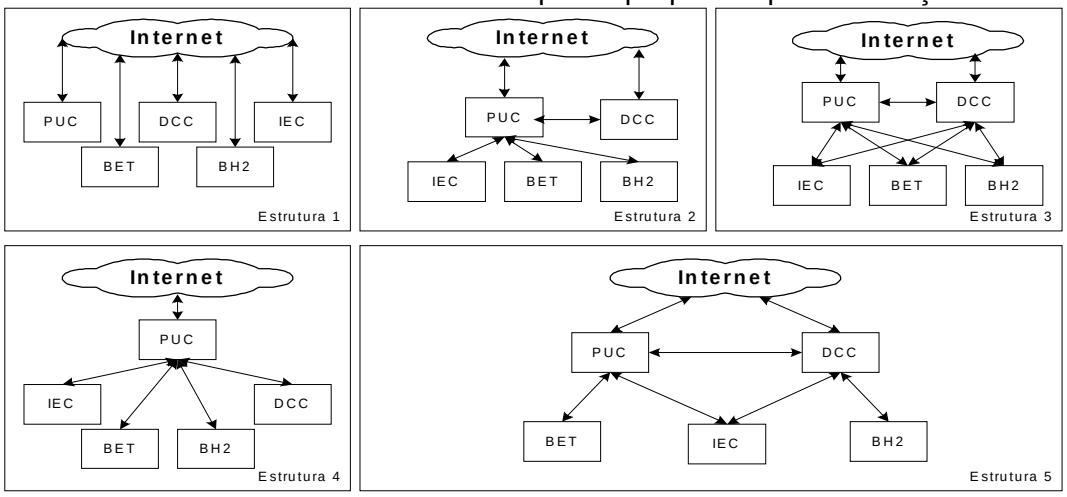
A Estrutura 4 caracterizasse por manter um único servidor *Web-caching* pai, Cache PUC, e todos os outros quatro servidores serão filhos, Cache DCC, IEC, Betim e BH2. O motivo da seleção dessa estrutura é verificar os efeitos da

centralização dos erros gerados nos servidores *web-caching* filhos em um único servidor *web-caching* pai.

A Estrutura 5 será formada pelos 5 servidores dispostos em dois níveis, um primeiro nível com os servidores Cache PUC e Cache DCC colaborando entre si como irmãos. No segundo nível temos o Cache IEC filho do Cache PUC e DCC, o Cache Betim filho do Cache PUC, e o cache BH2 filho do Cache DCC. Assim, cada servidor do primeiro nível terá dois servidores filhos. Esta estrutura foi definida com o intuito de equilibrar a carga de trabalho, balanceando a carga gerada por erro nos servidores *web-caching* filhos para os Cache PUC e DCC.

A FIG 4.18 apresenta logicamente como estarão dispostos os servidores *web-caching* em cada uma das estruturas hierárquicas propostas para análise.

FIGURA 4.18 - Estruturas hierárquicas propostas para avaliação.



Como definida no método, esta etapa se limita a escolher as estruturas que serão simuladas. Algumas orientações básicas descritas no Capítulo 3 podem auxiliar nesta etapa, mas o principal aspecto que contribui neste momento é a experiência do pessoal que está executando a análise de desempenho.

4.3.2 Simulação (E9)

Considerando o objetivo desta etapa mencionado no Capítulo 3, devemos preparar o ambiente laboratorial para a execução das simulações usando os modelos obtidos em etapas anteriores.

Devido a alta variabilidade nos resultados apresentados pelo aplicativo escolhido para executar as simulações e a orientação do Capítulo 3 que sugere repetir mais de uma vez cada simulação com as mesmas características, resolvemos trabalhar com a média ponderada recolhida de três simulações cujos *hit* em percentagem de volume de dados (HRKB) não extrapolassem os 5% definidos como margem de erro aceitável. A margem de erro foi verificada nos servidores que só atendiam requisições dos processos clientes.

As simulações foram executadas utilizando o diagrama apresentado na FIG. 4.17 e o segundo modelo de carga utilizado na etapa de validação e calibração do modelo de carga (E6).

Adotamos três simulações para cada estrutura, procurando reduzir erros provocados por distorção em uma experimentação. Se trabalhássemos com a análise sendo feita sobre uma única simulação, estaríamos sujeitos aos 5% de erro estabelecidos como margem aceitável. Definir mais simulações causaria uma elevação no tempo total necessário para executar todos os ensaios, porém a expectativa era que o erro acumulado fosse tendendo a zero quanto maior o número de experimentos. Ou seja, a tendência é que a média de dez experimentos classificados como aceitáveis tivessem uma margem de erro menor do que três experimentos.
Depois de executadas todas as simulações, nós armazenamos os resultados obtidos e extraímos os valores dos parâmetros definidos no início de nosso estudo de caso (E3). Estes valores serão apresentados em tabelas condensadas dentro dessa sessão, juntamente com uma apresentação gráfica e uma análise dos resultados.

Na apresentação das tabelas e dos gráficos os parâmetros utilizados são aqueles que descrevem de que forma os servidores de *web-caching* recuperaram os objetos que não estavam em sua área de *web-cache* local. Estes parâmetros são apresentados em função do volume de dados e quantidade de requisições e são classificados entre as três categorias: *hit* no pai ou irmão; recuperado direto da origem; e atendido pelo pai. As taxas são calculadas dentro do universo de requisições que causaram erro na base de dados local e não no total de requisições atendidas pelo servidor.

4.3.2.1. Simulação Estrutura 1

A Estrutura 1 mantém uma distribuição única em relação a maneira como requisições que geraram erro são atendidas pelos servidores *web-caching*. Não existindo nenhuma colaboração entre os servidores, toda requisição que gerou *miss* na base local de dados provocou uma consulta direta à origem, ou seja, em 100% dos casos, por isso não apresentamos sua distribuição.

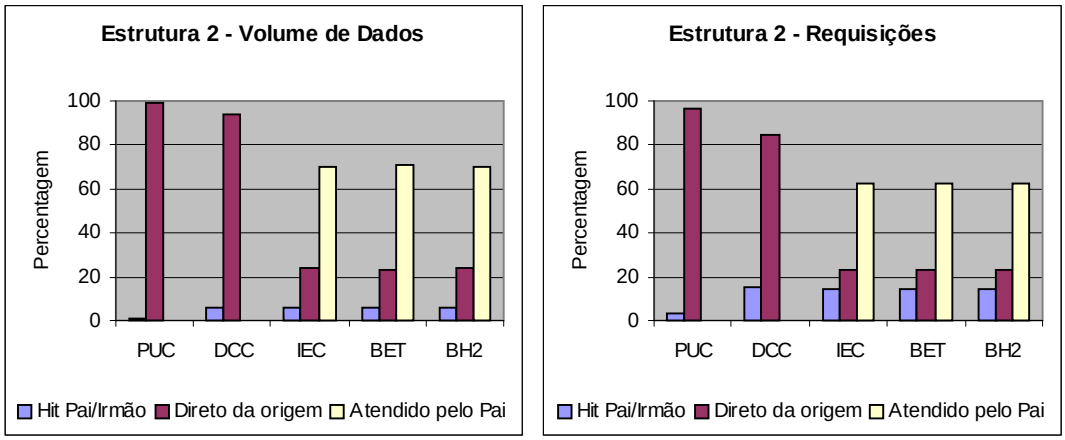
4.3.2.2. Simulação Estrutura 2

Comparando os resultados da TAB. 4.3 e 4.8 que apresentam os resultados da Estrutura 2 no ambiente real e simulado, respectivamente, em função da forma como os servidores *web-caching* recuperaram os objetos solicitados em caso de erro na base local do *web-cache*, podemos verificar que o comportamento dos servidores *web-caching* foram significativamente diferentes, o que permite questionar se o modelo de carga de trabalho formulado é adequado. Como já mencionamos, esta distorção está sendo provocada em parte pelas diversas simplificações impostas ao modelo de carga de trabalho.

TABELA 4.8 - Resultados da Simulação Estrutura 2

		Volume de Dados			Requisições		
		Hit pai/irmão	Direto da origem	Atendido pelo pai	Hit pai/irmão	Direto da origem	Atendido pelo pai
Cache PUC	1o. Sim.	1,14	98,86	0,00	3,24	96,76	0,00
	2o. Sim.	1,23	98,77	0,00	3,40	96,60	0,00
	3o. Sim.	1,17	98,83	0,00	3,31	96,69	0,00
	Média	1,18	98,82	0,00	3,32	96,68	0,00
Cache DCC	1o. Sim.	6,03	93,97	0,00	15,67	84,33	0,00
	2o. Sim.	6,54	93,46	0,00	15,13	84,87	0,00
	3o. Sim.	6,11	93,89	0,00	15,04	84,96	0,00
	Média	6,23	93,77	0,00	15,28	84,72	0,00
Cache IEC	1o. Sim.	5,95	24,39	69,66	14,45	22,98	62,57
	2o. Sim.	6,28	24,96	68,77	14,46	23,63	61,91
	3o. Sim.	6,09	23,19	70,72	14,98	23,00	62,01
	Média	6,11	24,18	69,72	14,63	23,20	62,16
Cache BET	1o. Sim.	5,76	24,12	70,13	14,28	23,71	62,01
	2o. Sim.	6,62	23,23	70,15	15,09	22,92	61,99
	3o. Sim.	5,88	22,04	72,08	14,45	23,54	62,00
	Média	6,09	23,13	70,79	14,61	23,39	62,00
Cache BH2	1o. Sim.	5,86	23,36	70,78	14,42	23,69	61,90
	2o. Sim.	5,90	25,04	69,06	14,52	23,24	62,24
	3o. Sim.	6,10	23,15	70,75	14,87	22,59	62,54
	Média	5,95	23,85	70,20	14,60	23,17	62,23

FIGURA 4.19 - Distribuição resultados coletados da Estrutura 2.



Devemos observar que validamos o modelo de carga de trabalho em função do acerto local nos servidores *web-caching* e não na forma como eles recuperaram objetos decorrentes de erros no *web-cache* local.

Observando as FIG. 4.15 e 4.19 fica mais clara a distorção apresentada nestes parâmetros referentes à forma como os servidores de *web-caching* recuperaram objetos não armazenados localmente.

A princípio o Cache PUC manteve seu comportamento semelhante ao ambiente real, ou seja, uma grande maioria dos objetos recuperados direto da origem e pouca colaboração com seu servidor irmão o Cache DCC que, por sua vez, já obteve melhor resultado de cooperação com o Cache PUC, cooperação esta igual para todos os outros servidores. Este comportamento padrão é justificado pela configuração do processo cliente que produzia as requisições aos servidores com taxas de acertos simétricas.

Outra característica que podemos observar é que a participação do Cache PUC no atendimento as requisições dos servidores filhos foi bem maior do que a observada no ambiente real. Esta característica provavelmente deve-se ao fato de não simularmos o atraso causado pela limitação do canal de comunicação entre as unidades. Com isso o Cache PUC atendia os servidores

filhos dentro do tempo de vida das requisições, evitando consultas diretas à origem por parte dos servidores filhos.

Havíamos mencionado anteriormente (item 4.2.5.6) a necessidade de aumentar o tempo de vida das requisições dos servidores filhos ao servidor pai no ambiente em produção, procurando assim, acumular melhores resultados de colaboração da estrutura hierárquica.

A distorção dos valores observada entre a TAB. 4.3 e 4.8 seria suficiente para justificar uma calibração no modelo de carga de trabalho para produzir resultados de *hit* em percentagem tanto de requisições como em volume de dados semelhantes nos dois ambientes, mas lembramos que fica inviável uma calibração no modelo devido a série de limitações detectadas na ferramenta de simulação escolhida, por este motivo não estaremos procedendo com esta calibração.
Resumindo, as simplificações impostas ao modelo de carga de trabalho produziram efeitos substancias nos resultados obtidos no ambiente simulado que poderiam ser contornados com um modelo de carga mais representativo. Vale ressaltar que estes efeitos agem sobre todas as estruturas avaliadas podendo beneficiar ou não alguma delas. Portanto é difícil garantir neste momento que a seleção da melhor estrutura para este estudo de caso será prejudicada.

Mais uma vez, afirmamos que esta diferença de resultados apresentados entre valores do ambiente real e simulado não invalidam nosso método, apenas o modelo de carga de trabalho utilizado.

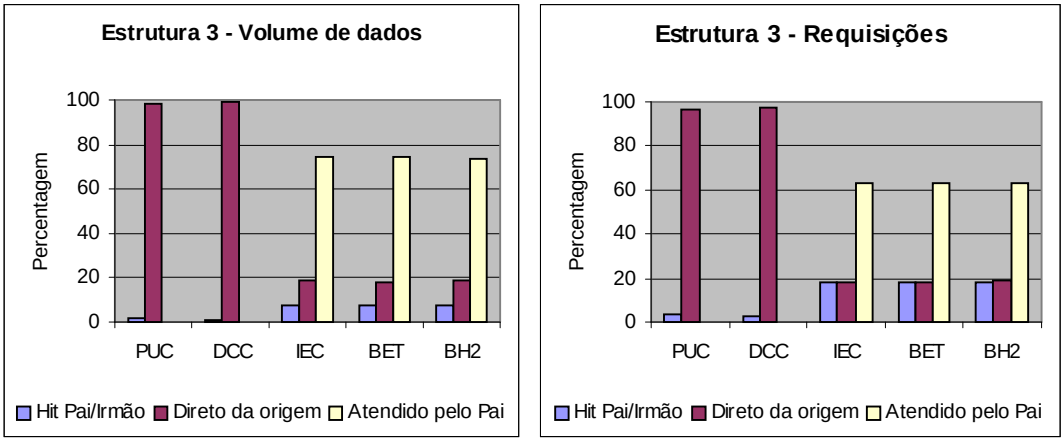
4.3.2.3. Simulação Estrutura 3

A TAB. 4.9 apresenta os valores obtidos durante a simulação da estrutura hierárquica de servidores *web-caching* disposta de acordo com a Estrutura 3. A seguir é apresentado a FIG. 4.20 que ilustra os resultados obtidos. Logo após, é feita uma análise rápida dos resultados.

TABELA 4.9 - Resultados da Simulação Estrutura 3

		Volume de Dados			Requisições		
		Hit pai/irmão	Direto da origem	Atendido pelo pai	Hit pai/irmão	Direto da origem	Atendido pelo pai
Cache PUC	1o. Sim.	1,31	98,69	0,00	3,52	96,48	0,00
	2o. Sim.	1,37	98,63	0,00	3,60	96,4	0,00
	3o. Sim.	1,44	98,56	0,00	3,64	96,36	0,00
	Média	1,37	98,63	0,00	3,59	96,41	0,00
Cache DCC	1o. Sim.	1,08	98,92	0,00	2,70	97,3	0,00
	2o. Sim.	1,12	98,88	0,00	2,64	97,36	0,00
	3o. Sim.	0,95	99,05	0,00	2,59	97,41	0,00
	Média	1,05	98,95	0,00	2,64	97,36	0,00
Cache IEC	1o. Sim.	7,23	17,12	75,65	18,12	18,31	63,58
	2o. Sim.	7,85	18,33	73,81	18,31	18,33	63,36
	3o. Sim.	7,61	19,25	73,14	18,28	18,24	63,48
	Média	7,56	18,23	74,20	18,24	18,29	63,47
Cache BET	1o. Sim.	7,66	18,24	74,11	18,11	18,67	63,22
	2o. Sim.	7,90	17,52	74,57	18,44	17,99	63,57
	3o. Sim.	7,70	17,88	74,42	18,37	18,33	63,30
	Média	7,75	17,88	74,37	18,31	18,33	63,36
Cache BH2	1o. Sim.	7,53	19,03	73,44	18,18	18,81	63,01
	2o. Sim.	7,90	18,76	73,34	18,14	18,41	63,44
	3o. Sim.	7,42	18,24	74,34	18,01	18,69	63,31
	Média	7,62	18,68	73,71	18,11	18,64	63,25

FIGURA 4.20 - Distribuição resultados coletados da Estrutura 3.



Na Estrutura 3 os servidores filhos Cache IEC, Betim e BH2 possuem taxas de acerto nos servidores pais Cache PUC e DCC maiores do que na Estrutura 2. Além disso, os servidores pais atendem um volume maior de requisições que resultaram em *miss* nos *web-cache* locais dos servidores filhos, reduzindo a necessidade de consultas direto à origem dos documentos por parte dos servidores filhos.

Os servidores Cache PUC e DCC apresentam baixa colaboração entre si. Ambos trabalham com grande quantidade de transações e isto talvez tenha comprometido a capacidade de processamento, causando atraso na comunicação entre os servidores irmãos e ultrapassando o tempo de vida das requisições. Afirmamos isto porque durante as experimentações acompanhamos que algumas poucas requisições não foram atendidas por sobrecarga na estrutura hierárquica.

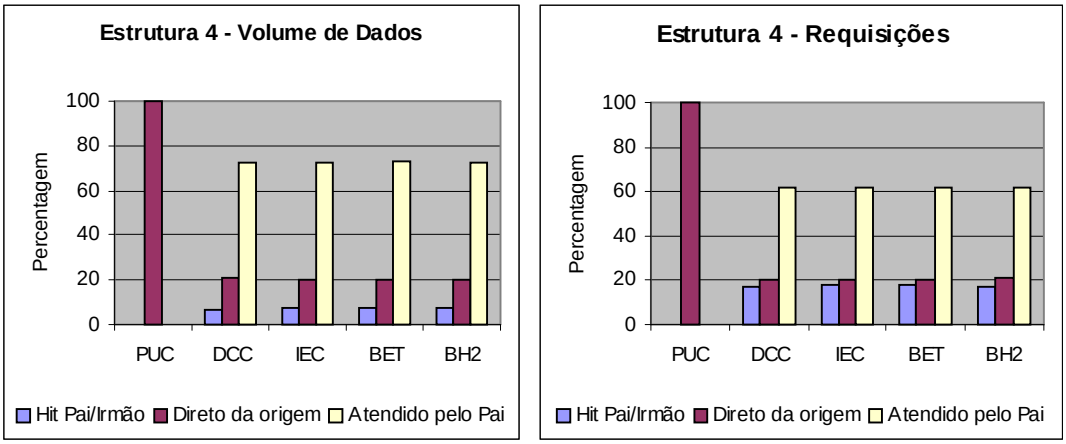
4.3.2.4. Simulação Estrutura 4

A TAB. 4.10 apresenta os valores obtidos durante a simulação da estrutura hierárquica de servidores *web-caching* disposta de acordo com a Estrutura 4. A seguir é apresentado a FIG. 4.21 que ilustra os resultados obtidos. Logo após, é feita uma análise rápida dos resultados.

TABELA 4.10 - Resultados das Simulações Estrutura 4

		Volume de Dados			Requisições		
		Hit pai/irmão	Direto da origem	Atendido pelo pai	Hit pai/irmão	Direto da origem	Atendido pelo pai
Cache PUC	1o. Sim.	0,00	100,00	0,00	0,00	100	0,00
	2o. Sim.	0,00	100,00	0,00	0,00	100	0,00
	3o. Sim.	0,00	100,00	0,00	0,00	100	0,00
	Média	0,00	100,00	0,00	0,00	100,00	0,00
Cache DCC	1o. Sim.	6,77	22,61	70,62	17,39	20,67	61,95
	2o. Sim.	7,31	19,25	73,44	17,42	20,39	62,19
	3o. Sim.	7,02	21,03	71,95	17,25	20,58	62,17
	Média	7,03	20,96	72,00	17,35	20,55	62,10
Cache IEC	1o. Sim.	6,93	19,81	73,26	17,41	20,24	62,35
	2o. Sim.	7,50	20,13	72,38	17,56	20,61	61,83
	3o. Sim.	7,31	21,08	71,61	17,41	20,43	62,16
	Média	7,25	20,34	72,42	17,46	20,43	62,11
Cache BET	1o. Sim.	7,05	19,82	73,12	17,46	20,43	62,11
	2o. Sim.	7,50	21,19	71,32	17,48	20,88	61,65
	3o. Sim.	7,17	18,51	74,32	17,39	20,39	62,21
	Média	7,24	19,84	72,92	17,44	20,57	61,99
Cache BH2	1o. Sim.	7,22	19,09	73,69	17,23	20,88	61,88
	2o. Sim.	7,51	20,45	72,04	17,23	20,86	61,91
	3o. Sim.	7,39	21,19	71,42	17,32	20,65	62,03
	Média	7,37	20,24	72,38	17,26	20,80	61,94

FIGURA 4.21 - Distribuição resultados coletados da Estrutura 4.



Nesta simulação o Cache PUC chega a receber uma carga duas vezes e meio maior que o definido em seu processo cliente respectivo. Isto se deve a estrutura hierárquica projetada que direciona todo erro gerado nos servidores *web-caching* filhos ao Cache PUC. Mesmo assim, o servidor consegue atender de forma satisfatória os servidores filhos.

Todos os servidores filhos têm um aproveitamento uniforme em relação às consultas redirecionadas ao servidor pai, reduzindo substancialmente a necessidade de consulta à origem dos objetos.

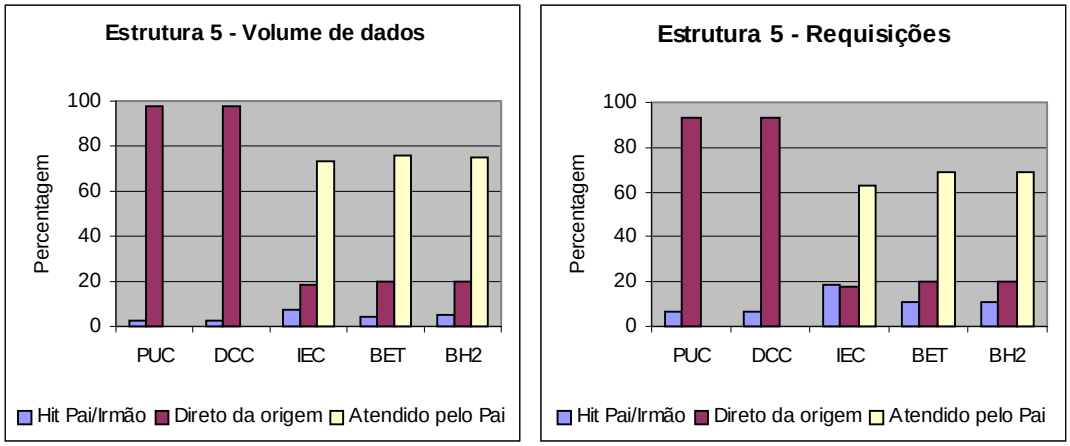
4.3.2.5. Simulação Estrutura 5

A TAB. 4.11 apresenta os valores obtidos durante a simulação da estrutura hierárquica de servidores *web-caching* disposta de acordo com a Estrutura 5. A seguir é apresentado a FIG. 4.22 que ilustra os resultados obtidos. Logo após, é feita uma análise rápida dos resultados.

TABELA 4.11 - Resultados das Simulações Estrutura 5

		Volume de Dados			Requisições		
		Hit pai/irmão	Direto da origem	Atendido pelo pai	Hit pai/irmão	Direto da origem	Atendido pelo pai
Cache PUC	1o. Sim.	2,80	97,20	0,00	6,68	93,32	0,00
	2o. Sim.	2,16	97,84	0,00	5,57	94,43	0,00
	3o. Sim.	2,66	97,34	0,00	7,00	93	0,00
	Média	2,54	97,43	0,00	6,42	93,58	0,00
Cache DCC	1o. Sim.	2,43	97,57	0,00	6,26	93,74	0,00
	2o. Sim.	2,88	97,12	0,00	7,58	92,42	0,00
	3o. Sim.	2,50	97,50	0,00	6,10	93,9	0,00
	Média	2,60	97,40	0,00	6,65	93,35	0,00
Cache IEC	1o. Sim.	8,13	18,41	73,45	18,39	18,36	63,25
	2o. Sim.	7,98	18,89	73,13	18,66	17,58	63,77
	3o. Sim.	7,61	18,40	73,99	18,68	18,17	63,15
	Média	7,91	18,57	73,52	18,58	18,04	63,39
Cache BET	1o. Sim.	4,58	19,68	75,74	10,34	20,31	69,35
	2o. Sim.	4,84	20,68	74,48	11,15	20,02	68,83
	3o. Sim.	4,26	18,84	76,89	10,42	20,21	69,37
	Média	4,56	19,73	75,70	10,64	20,18	69,18
Cache BH2	1o. Sim.	4,88	21,07	74,05	10,52	20,46	69,02
	2o. Sim.	4,44	19,13	76,43	10,27	20,26	69,47
	3o. Sim.	4,72	20,30	74,98	10,89	20,23	68,88
	Média	4,68	20,17	75,15	10,56	20,32	69,12

FIGURA 4.22 - Distribuição resultados coletados da Estrutura 5.



Na Estrutura 5 a carga é melhor distribuída entre o Cache PUC e Cache DCC, ambos dividem a função de recuperação de dados da origem (FIG. 4.18). O servidor Cache IEC por ter dois servidores pai tem uma taxa de acerto maior que o Cache BET e BH2 (FIG. 4.22) e conseqüentemente é um pouco menor a necessidade de obter dados da origem.

Os servidores irmãos já apresentam uma colaboração maior do que a apresentada na Estrutura 3, justificada em parte por uma menor carga de requisições que circulou na estrutura.

4.3.2.6 Observações gerais

Durante a execução dessa etapa foi possível observar algumas variações nos resultados em relação ao ambiente real causadas pelas simplificações realizadas no modelo de carga de trabalho e deficiências na ferramenta de simulação escolhida, mencionadas na etapa de validação e calibração (item 4.2.6).

Outro aspecto importante é o comportamento instável da ferramenta utilizada para simular o ambiente real. Muitas simulações foram descartadas devido a variação fora da margem de erro aceitável definida durante o estudo de caso. Isto provocou um grande atraso nesta etapa e inviabilizou uma nova calibração

no modelo de carga de trabalho que poderia minimizar as variações observadas e já mencionadas.

No geral o comportamento dos servidores permaneceu dentro do esperado, mas as variações observadas durante a simulação da Estrutura 2 merecem atenção especial.

A falta de recursos computacionais também fica mais evidente durante esta etapa e procuramos tomar o devido cuidado para evitar que esta limitação de recursos provocasse efeitos nas simulações.

Supondo que o modelo de carga simplificado represente a rede acadêmica da PUC-Minas, os resultados servem para validar esta etapa do método e comprova sua importância dentro do método proposto. Nela verificamos uma interação com várias etapas anteriores e a necessidade de um compromisso muito sério do responsável em realizar a análise de desempenho.

4.3.3 Seleção da Melhor Estrutura (E10)

Os três modelos (carga, desempenho e custo) definidos durante a fase de validação e calibração dos modelos devem ser considerados para selecionar a melhor estrutura hierárquica de *web-caching*. Em nosso estudo de caso, não estamos preocupados com o modelo de custo, ou seja, não discutiremos a viabilidade financeira para se implantar a melhor solução. Além disso, não exploramos experimentalmente, nenhuma estrutura que demandasse aquisição de novos recursos.

O modelo de carga apresenta suas limitações, mas atendeu as expectativas de promover carga no ambiente simulado, agora devemos aplicar o modelo de desempenho definido durante uma das etapas do método. Em nosso estudo de caso definimos que o modelo de desempenho é simples e trabalha com o somatório do volume de dados recuperados direto da origem. Na TAB. 4.12

apresentamos os resultados em volume de dados total (KB) recuperados direto da origem da informação para cada estrutura e em cada experimentação, assim como a média aritmética de cada servidor *web-caching* e cada estrutura, valor este que será usado no somatório. Na FIG 4.23 são apresentados os valores obtidos no somatório de cada estrutura simulada.

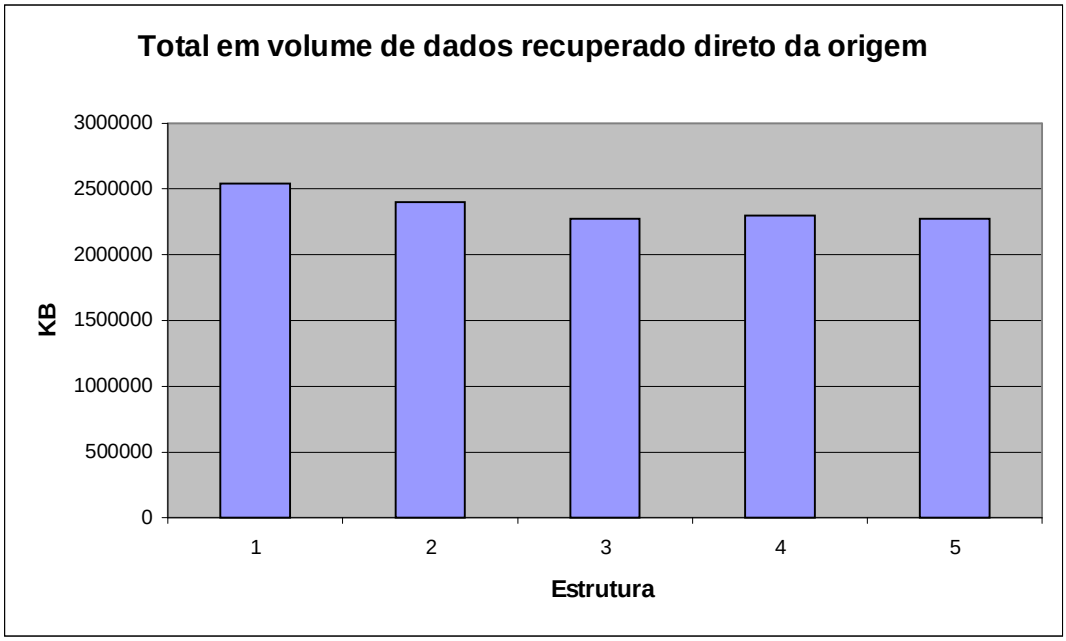
TABELA 4.12 - Valores de cada simulação e médios do volume de dados total recuperado da origem da informação.

Servidor		Estrutura 1	Estrutura 2	Estrutura 3	Estrutura 4	Estrutura 5
Cache PUC	1º Sim.	489923	1547911	980539	1884332	938111
	2º Sim.	533539	1511821	970043	1828340	1047791
	3º Sim.	521035	1567848	959992	1955178	986324
	Média	514832	1542527	970191	1889283	990742
Cache DCC	1º Sim.	509243	510704	1043941	122883	1016067
	2º Sim.	500746	456191	1049827	90205	959607
	3º Sim.	514831	498736	1022496	108816	993032
	Média	508273	488544	1038755	107301	989569
Cache IEC	1º Sim.	500413	124361	86690	104891	88834
	2º Sim.	517546	126045	88782	98692	92879
	3º Sim.	492984	120962	95648	104758	93087
	Média	503648	123789	90373	102780	91600
Cache BET	1º Sim.	503951	121718	89968	105393	92244
	2º Sim.	507799	117424	83171	101728	105590
	3º Sim.	497636	116725	85776	96080	92487
	Média	503129	118622	86305	101067	96774
Cache BH2	1º Sim.	512336	119976	94899	98170	99052
	2º Sim.	512862	130548	90409	98832	91792
	3º Sim.	499888	121218	90540	104567	96407
	Média	508362	123914	91949	100523	95750
Somatório das Médias		**2538244**	**2397396**	**2277574**	**2300955**	**2264435**

Observando os resultados e utilizando o modelo de desempenho definido no estudo de caso, a Estrutura 5 é escolhida como a que tem melhor desempenho para o ambiente da rede acadêmica da PUC-Minas, tendo em vista que o somatório das médias da Estrutura 5 apresentou o menor valor. A Estrutura 3 seria uma segunda opção com boa performance.

Esperávamos ter a Estrutura 3 como melhor estrutura em termos de desempenho, tal resultado não foi observado aparentemente por dois motivos, o primeiro devido a série de simplificações impostas ao modelo de carga e depois devido a uma elevada carga durante a simulação da Estrutura 3 que acreditamos ter provocado muitas consultas diretas à origem dos objetos por falta de respostas dos servidores *web-caching* dentro da hierarquia.

FIGURA 4.23 - Somatório dos resultados experimentados.



4.3.4 Validação em Produção (E11)

Esta etapa não foi executada explicitamente em nosso estudo de caso, por dois motivos que discutiremos a seguir.

O modelo de carga de trabalho do estudo de caso não é muito representativo e os resultados obtidos dificilmente representam um comportamento que seria verificado no ambiente real em produção. Esta distorção de comportamento entre ambiente real e simulado acarretaria o retorno para etapa de definição do

modelo de carga de trabalho e atrasaria o cronograma do projeto. Esta possibilidade é prevista no método.

O outro motivo está relacionado ao cronograma da elaboração da dissertação que não previa a implantação de uma estrutura hierárquica com um período longo de acompanhamento. Acreditamos que os resultados que seriam obtidos atingiriam o objetivo de comprovar a eficiência do método proposto, inclusive a etapa E11.

Esta etapa pode ser validada supondo que a estrutura selecionada na etapa E10 fosse a Estrutura 2, podemos considerar o período de coleta de dados onde a estrutura hierárquica esteve definida de acordo com a Estrutura 2 como sendo a etapa de validação em produção, chegaríamos a conclusão que os resultados estariam fora de uma margem de erro aceitável exigindo retorno ao início do método para ajustes no modelo de carga de trabalho e/ou na fase de experimentação.

Considerando esta distorção voltamos ao estado da arte e analisando LIDEMANN [49] percebemos que a ferramenta de análise de arquivos de eventos escolhida não era a mais indicada, tendo em vista que ela não extrai dos arquivos de eventos informações como: tipos de objetos acessados, tamanho médio, popularidade, distribuição, *cachable*, etc. Informações estas que deveriam conduzir de forma mais adequada à formulação do modelo de carga de trabalho.

Portanto, podemos enumerar três fatores que provocaram tal distorção: falta de recursos computacionais para simular ambiente em laboratório; simplificações aplicadas no modelo de carga de trabalho para facilitar e reduzir os tempos de simulações; definição de um modelo de carga não muito representativo devido à seleção errada da ferramenta de análise de eventos.

4.4 Agentes

Como o método identifica, nosso estudo de caso também pode ser enquadrado na divisão dos três grupos que interagem diretamente com o problema sob estudo. Cabe-nos agora verificar quais são os integrantes de cada um dos grupos e discutir suas participações na escolha ou seleção da melhor estrutura.

O grupo usuários é representado, principalmente, por alunos e professores. Os alunos formam a grande maioria dentro deste grupo. Além disso, podem ser considerados como usuários muito exigentes, tendo em vista que pagam mensalidades e se vêem no direito de ter alta qualidade nos serviços disponibilizados. Em compensação, uma fatia considerável desses alunos não coopera na ocupação dos recursos, utilizando-os para fins não apropriados e prejudicando pesquisas realizadas por outra parte dos usuários. Por exemplo, *download* de arquivos mp3, navegação em sites de conteúdo impróprio, *download* de arquivos grandes não programados disparados em horários de grande demanda dos recursos, etc.

Outro grupo é a equipe técnica formada basicamente por técnicos que tem como função manter os servidores de *web-caching* em funcionamento e se encarregar de aplicar restrições de uso de acordo com demanda apresentada pelos professores. Os professores têm condição de retirar a Internet do laboratório onde haverá aula evitando que os alunos se desviem das atividades planejadas durante aquele período.

Além dos técnicos, existe também um analista responsável em acompanhar o comportamento dos servidores de *web-caching*, assim como propor novas configurações com o intuito de melhorar o desempenho geral de acesso. Estas soluções podem estar relacionadas a pequenas mudanças de configuração nas máquinas e/ou nos *softwares*, como também indicação para aquisição de recursos mais específicos, por exemplo, *link* Internet, novos servidores, etc.

O terceiro grupo em nosso estudo de caso é a gerencia administrativa da instituição, em particular a Pró-Reitoria de Infra-Estrutura que coordena a aquisição de novos recursos e procura distribuir recursos financeiros de acordo com a severidade dos problemas ocorridos em toda a organização.
Os conflitos são naturais. As disputas pelos recursos são freqüentes e a demanda por serviços que consomem mais banda de comunicação é cada vez mais comum. É inevitável a necessidade de investimentos constantes nesta área, nem sempre possíveis.

A etapa em si nos alerta a possibilidade de escolhermos a estrutura não só baseado nos resultados do modelo de desempenho, mas sim por um conjunto de fatores. Em nosso estudo de caso, como todas as estruturas escolhidas para simulação demandam dos mesmos recursos o que pode ser questionado é a atualização do canal Internet e dos canais de comunicação para as unidades.

4.5 Conclusões

Seguindo todas as etapas delineadas em nosso método conseguimos, de forma organizada, implementar todo o processo que nos levou a uma completa análise de desempenho de estruturas hierárquicas de servidores *web-caching*.

O estudo de caso teve início com o entendimento do ambiente e com o estudo e escolha da implementação de *web-caching* que seria utilizada. Depois definimos quais seriam os parâmetros que estariam em análise e em seguida escolhemos as ferramentas que possibilitariam todas as simulações e análise de resultados. A etapa de coleta de dados foi bastante aprofundada gerando um excelente banco de dados e deixando claro qual poderia ser o ganho no desempenho com o estabelecimento de cooperação entre os servidores de *web-caching*. Com a execução de todas estas etapas já tínhamos definido um modelo de carga de trabalho e de desempenho que sofreram ajustes até serem

validados, apesar das simplificações realizadas. Neste momento já havíamos encerrado a fase 1 de caracterização e validação dos modelos.

Ao iniciarmos a segunda fase, de experimentação, definimos quais estruturas seriam simuladas e avaliadas, respeitando algumas orientações propostas no método. Com as estruturas definidas partimos para a etapa de simulação de cada uma, quando além de simularmos fomos acumulando os resultados usados logo em seguida na etapa de seleção da melhor estrutura.

Todas as etapas foram seguidas de acordo com o método proposto, assim como as relações entre as etapas e modelos podem ser acompanhadas durante todo este capítulo que apresentou o estudo de caso da rede acadêmica da PUC-Minas.

Podemos verificar que são diversos os parâmetros que podem ser avaliados e considerados durante a especificação de uma análise de desempenho de estruturas hierárquicas de servidores de *web-caching*. Cabem diversos estudos que verifiquem a influência de cada um dos componentes e parâmetros que formam o ambiente como um todo.

Neste nosso estudo de caso, foi possível concluir que a modelagem da carga de trabalho é uma tarefa muito árdua. É necessário estar bem fundamentado para a escolha de valores quantitativos para as variáveis que formam o modelo. Qualquer simplificação pode gerar grandes oscilações no comportamento do ambiente simulado que, provavelmente, só serão verificados em uma validação em produção. A escolha de uma boa ferramenta de simulação e análise de eventos também influenciará em muito a definição do modelo de carga. Como foi observado diversas vezes durante este capítulo, a ferramenta selecionada para nosso estudo de caso apresentou algumas limitações prejudicando os resultados e etapas da fase de experimentação. Uma opção para contornar este problema é utilizar uma ferramenta de simulação que utilize arquivos de eventos reais para promover carga em um

ambiente simulado. Dessa forma, o modelo de carga seria fiel ao ambiente real, reduzindo bastante falhas na fase de experimentação.

A própria escolha, em nosso estudo de caso, da Estrutura 5 pode estar sujeita ou vinculada a decisões erradas tomadas no início de nossa modelagem. Um exemplo disso é o limite de capacidade de processamento atingido durante as simulações que provocou inclusive o não atendimento de algumas requisições.

Muitas incógnitas permanecem, principalmente as que se referem ao comportamento das simulações se um ou outro argumento tivesse sido alterado ou considerado.

Resumindo, a aplicação do método proposto em um estudo de caso real teve sucesso o que comprova a eficiência de nossa proposta. Porém a análise de desempenho em laboratório, como mencionado no Capítulo 3, tende a ser demorada e complexa.

CAPÍTULO 5 - CONCLUSÃO

5.1. Discussão Geral

O escopo inicial dessa pesquisa era tratar a situação particular vivida por uma instituição. Nosso objetivo seria aplicar um método para selecionar uma estrutura hierárquica de servidores *web-caching* que atendesse, com o melhor desempenho, a demanda gerada pelos usuários desta instituição.

Durante os estudos verificamos a falta de um método específico para atender nossa necessidade. Nesse momento resolvemos propor um método cujo objetivo é auxiliar a análise de desempenho de estrutura hierárquica de servidores *web-caching*, permitindo a escolha de uma estrutura que atenda as organizações da melhor forma possível. Este método foi formulado para possibilitar análise a partir de experimentações em laboratório, tendo em vista a dificuldade, na maioria dos casos, em desenvolver um modelo analítico e no elevado tempo e transtorno operacional para realizar medição ou monitoramento em ambiente real.

Apesar da proposta de método apresentada para análise de desempenho de servidores *web-caching* seguir o paradigma do método de análise de desempenho de estruturas cliente/servidor, nosso método considera particularidades que acreditamos contribuir significativamente com as pesquisas em torno da técnica de *web-caching*. Além disso, a aplicação do método a um estudo de caso exemplifica bem a capacidade do método e sua eficiência.

O método foi baseado em modelos, etapas, sub-etapas e relacionamentos. Apresentamos no Capítulo 3 todas as características de cada etapa e da formulação dos modelos exceto do modelo de custo. Cada um dos relacionamentos foi explicitado no texto com referências numéricas que auxiliam o acompanhamento do texto junto com a FIG. 3.1. Durante a

conclusão do Capítulo 3 apresentamos o que poderia ser um método simplificado para análise de desempenho em algumas situações.

Ao enumerarmos todos os componentes do método visualizamos a possibilidade de automatização do processo de análise de desempenho de estruturas hierárquicas de servidores *web-caching* que se transformaria em uma poderosa ferramenta de análise e suporte aos pesquisadores e analistas.

Com o método definido, devíamos verificar a correção de todos os seus componentes conseqüentemente comprovando a eficiência do método como um todo. Para tanto escolhemos como estudo de caso a rede acadêmica da PUC-Minas para proceder com esta avaliação.

No Capítulo 4 são discutidas todas as razões que nos motivaram a escolher a rede acadêmica da PUC-Minas para o estudo de caso. O importante desse estudo de caso é verificar se a aplicação do método em uma situação real produz o resultado esperado.

O estudo de caso teve uma duração bastante longa - oito meses. Este longo período foi decorrente das etapas de coleta de dados e de simulação estendidas ao máximo para permitir uma boa modelagem da carga de trabalho. Além disso, tomamos o cuidado de apresentar quantitativamente todos os parâmetros trabalhados e justificar muitas decisões tomadas.

Os principais problemas enfrentados nessa dissertação aconteceram durante a aplicação do método proposto no estudo de caso selecionado. Primeiramente decidimos formular o modelo de carga de trabalho seguindo características mais próximas ao ambiente em estudo. Para isso tivemos que projetar e implementar uma estrutura de servidores de *web-caching* que nos permitisse acumular dados de comportamento dos usuários. O tempo de coleta também não poderia ser pequeno para evitar comprometimento dos arquivos de eventos armazenados.

Após este período inicial de algumas semanas verificamos ainda que não tínhamos conhecimento do comportamento da estrutura a partir do momento que esta trabalhasse de forma cooperativa. Resolvemos então aplicar uma estrutura hierárquica simples no ambiente sob estudo para verificar novamente o comportamento. Esta segunda coleta de dados durou mais 45 dias.

Com o término da etapa de coleta de dados tínhamos um grande volume de eventos do comportamento dos usuários de nosso ambiente real e, conseqüentemente, condições de estabelecer o modelo de carga para o estudo de caso. O problema seguinte foi montar em laboratório, com limitações de recursos computacionais, uma estrutura que simulasse o ambiente real. Esta limitação nos fez estudar alternativas para simular a estrutura com poucos recursos.

O modelo de carga de trabalho foi formulado aplicando-se algumas simplificações com o objetivo de reduzir o tempo de experimentação, mas que implicou em perda de representatividade do ambiente real e conseqüente distorção observada na etapa de validação em produção. Limitações na ferramenta selecionada para simulação também promoveram certas distorções. No entanto, nosso objetivo não era identificar a verdadeira estrutura com melhor desempenho para o estudo de caso selecionado e sim verificar se o método permite tal conclusão o que foi comprovado.

Com isso, nossas etapas de coleta de dados, montagem do ambiente de simulação e a experimentação duraram vários meses, atrasando o cronograma estabelecido.

5.2 Análise dos Objetivos, Metas e Resultados.

Apresentamos no Capítulo 1 as dificuldades e problemas enfrentados com o surgimento da Internet e no Capítulo 2 indicamos uma solução adotada por muitas entidades para minimizar estes problemas. Esta solução considera o

uso de servidores *web-caching*, porém também apresenta vantagens e desvantagens o que permite a discussão de diversos aspectos distintos. Nossa preocupação foi trabalhar a propriedade que possibilita cooperação entre diversos servidores *web-caching*, com o objetivo de aumentar a taxa de acertos tanto em requisições como em volume de dados dentro da estrutura de servidores refletindo em melhora de desempenho e aproveitamento de recursos.

Analisando os resultados apresentados na etapa de escolha da melhor estrutura hierárquica no Capítulo 4 podemos verificar como a propriedade de cooperação entre os servidores de *web-caching* é importante e qual é o ganho que se pode ter aplicando esta característica. Durante esta etapa apresentamos quantitativamente resultados observados em diferentes estruturas hierárquicas de servidores *web-caching*, o que possibilitou a escolha da estrutura mais eficiente para o estudo de caso selecionado.

Nossos resultados são importantes por apresentarem um método que orienta o processo de execução de análise de desempenho de estruturas hierárquicas de servidores *web-caching*, este método ao ser aplicado permite a escolha de ambientes com maiores benefícios para as entidades. Como benefícios podemos enumerar: melhor aproveitamento de recursos computacionais; redução da latência nas respostas aos usuários; aumento de disponibilidade de objetos, etc.

Em relação aos objetivos e metas dessa dissertação podemos afirmar que todos os objetivos foram alcançados. O objetivo principal era apresentar um método que auxiliasse a análise de desempenho de estruturas hierárquicas de servidores *web-caching*. Este método foi desenvolvido e descrito no Capítulo 3. Além da formalização do método feita no Capítulo 3 procedemos com a comprovação de eficiência do mesmo, aplicando-o a um estudo de caso.

Os outros objetivos definidos no Capítulo 1 são conseqüência direta da escolha e implantação da melhor estrutura hierárquica de servidores *web-caching* indicada pelo método proposto. Esses objetivos eram: otimização dos recursos, redução na latência nas respostas para os usuários; aumento na disponibilidade das informações e controle de acesso. Estes objetivos podem ser alcançados usando a estrutura hierárquica escolhida.

5.3 Conclusão Geral

Apresentamos conclusões parciais ao término das principais etapas de nossa pesquisa, sendo elas a apresentação do estado da arte (Capítulo 2), método propostos de análise de desempenho (Capítulo 3) e estudo de caso para verificação da eficiência do método (Capítulo 4). Dessa forma, os resultados finais de cada capítulo da dissertação podem ser analisados assim que foram apresentados.

No geral podemos concluir que a pesquisa atingiu seu objetivo. Definimos um contexto de trabalho onde explicitamos a área de abrangência de nossa pesquisa e quais os motivos que nos encaminharam para a escolha desse assunto.

Após a definição do contexto e do problema a ser pesquisado prosseguimos com o estudo do estado da arte de uma técnica amplamente utilizada na Internet. Este estudo avalia vários aspectos da técnica e é apresentado no Capítulo 2.

Com o enfoque da dissertação definido e com a apresentação da fundamentação da pesquisa partimos para a avaliação de métodos genéricos de análise de desempenho e apresentamos um método direcionado ao nosso contexto derivado de métodos genéricos. Este método é discutido por completo no Capítulo 3 e em seguida é aplicado a um estudo de caso com o intuito de verificar o método proposto.

5.4. Contribuições

O método de análise de desempenho de estruturas hierárquicas de servidores *web-caching* é de grande valia e pode ser considerado como uma contribuição importante de nossa pesquisa, já que não identificamos trabalhos similares durante o estudo do estado da arte. Este método é apresentado em detalhes e procuramos deixar claro cada passagem. O estudo de caso, além de verificar o método, indica as vantagens da utilização de servidores de *web-caching*.

Podemos identificar outras contribuições também significativas:

- A análise crítica dos aplicativos de suporte disponíveis para ensaios laboratoriais, tanto as ferramentas de análise de eventos como as de simulação de clientes e servidores *web*;
- A solução que simula vários servidores de *web-caching* com a utilização de apenas dois computadores, viabilizando experimentações laboratoriais;
- A apresentação de resultados quantitativos que podem servir para outras pesquisas;
- A discussão da participação do componente “Agente” durante a escolha da melhor solução para uma entidade.

É importante também observar que este é o primeiro trabalho de mestrado na área de *web* dentro do Programa de Pós-graduação de Engenharia Elétrica da PUC-Minas, criando um contexto e motivação para pesquisas futuras dentro do Programa.

5.5 Apresentação dos Principais Trabalhos Futuros

Como sugestão para trabalhos futuros podemos indicar uma série de análises e pesquisas mais aprofundadas em situações que provocaram certo questionamento na aplicação do método proposto durante o estudo de caso. Entre os vários possíveis trabalhos futuros podemos destacar alguns como:

A verificação detalhada ou completa do método proposto para determinar sua abrangência e confirmar se toda situação de análise de desempenho baseado na experimentação em laboratório está contemplada em nosso trabalho, ou seja, pesquisar possíveis limitações. Confirmada sua abrangência seria interessante a implementação em *software* do método para automatizar sua aplicação e agilizar a tarefa de análise de desempenho.

Aplicar ao estudo de caso selecionado um modelo de carga mais representativo, avaliando o impacto de algumas simplificações realizadas devido a escopo e tempo definidos nesta pesquisa.

Verificamos também a necessidade de uma ferramenta mais poderosa para simulação em laboratório. Observamos que as ferramentas que simulam as requisições dos clientes e os servidores de *web* utilizadas para promover carga de trabalho em um servidor *web-caching* são de difícil adaptação para promover carga de trabalho em uma estrutura hierárquica de servidores de *web-caching*. Talvez uma alternativa seja a realização de simulação em laboratório com arquivos de eventos reais.

Em relação à influência de parâmetros no comportamento dos servidores de *web-caching* acreditamos que se justifica a pesquisa nos seguintes assuntos: relação de área em disco reservada para armazenamento e volume de dados trafegados por unidade de tempo; capacidade de processamento dos servidores *web-caching*; e a influência do tempo de vida das requisições solicitadas aos servidores cooperados.

BIBLIOGRAFIA

[1] AGGARWAL, C., WOLF, J.L., YU, P.S., *Caching on the World Wide We***b**, IEEE Transactions on Knowledge and Data Engineering, Janeiro-Fevereiro 1999, Volume: 11, 14p.

[2] ALMEIDA, Jussara, CAO, Pei, *Measuring Proxy Performance with the Wisconsin Proxy Benchmark*, IN Third International WWW Caching Workshop, Manchester, Inglaterra, Junho, 1998.

[3] ALMEIDA, Virgilio A.F., BESTRAVOS, A., CROVELLA, M.Oliveira, *Characterizing reference locality in the www*, In Proceedings Of IEEE Conf. On Parallel and Distributed Systems (PDIS'96), Miami Beach, Flórida, Dezembro 1996.

[4] ALMEIDA, Vírgilio A. F., CESÁRIO M. G., FONSECA, R., MEIRA, Wagner Jr., MURTA, Cristina D., *The influence of geographical and cultural issues on the cache proxy server workload*, In WWW7-Seventh International World Wide Web Conference, edited by Computer Networks and ISDN Systems, 1998, volume 30, 4p.

[5] ARLITT, M., JIN, T., *A workload characterization study of the 1998 World Cup Web site*, IEEE Network, Maio-Junho 2000, volume 14, 8p.

[6] ARLITT, M. F., WILLIAMSON, C. L., *Web Server workload characterization: the search for invariants*, In Proc. Of ACM SIGMETRICS, Philadelphia, PA, Abril 1996.

[7] ASAKA, T., MIWA, H., TANAKA, Y., *Distributed Web caching using hash-based query caching method*, Control Applications, 1999. Proceedings of the 1999 IEEE International Conference on, 1999, volume 2, 26p.

[8] BANGA G., DRUSCHEL P., *Measuring the Capacity of a Web Server*, Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, California, Dezembro 1997.

[9] BARBOSA, Marco A, *Estatísticas do canal de comunicação Internet*, PUCMINAS, http://www.inf.pucmg.br/crc/servicos/estatisticas/

[10] BARISH, G., OBRACZKE, K., *World Wide Web caching: trends and techniques*, IEEE Communications Magazine, Volume 38, Capítulo 5, Maio 2000, 7p.

[11] BEERMANN, Cord, *Calamaris*, http://cord.de/tools/squid/calamaris/.

[12] BELFORD P., CROVELLA M., *Generating representative web workloads for network and server performance evaluation*, In Proceedings of the ACM SIGMETRICS Conference, Madison, WI, Julho, 1998.

[13] BESTRAVOS, A., WWW traffic reduction and load balancing through server-based caching, IEEE Concurrence, Janeiro-Março 1997, 10p.

[14] BRANDÃO, R.F., ANIDO, R.O., *A parallel simulator for distributed and cooperative web caches*, Fifth IEEE International Workshop on Distributed Simulation and Real-Time Applications, DS-RT 2001, 8p.

[15] BUSARI, Mudashiru, WILLIAMSON, Carey, *On the Performance of Heterogeneous Web Proxy Caching Hierarchies*, Department of Computer Science University of Saskatchewan, 1997.

[16] CAO, P., IRANI, S., *Cost-Aware WWW Proxy Caching Algorithms*, Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems, Dezembro, 1997, 14p.

[17] CAO, Pei, *Web Cache Simulator*, http://www.cs.wisc.edu/~cao/

[18] CECCHET, E. Parallel pull-based LRU: a request distribution algorithm for clustered Web caches using a DSM for memory mapped networks , Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid, 2001, 6p.

[19] CHE, Hao, WANG, Zhijung, TUNG, Ye, *Analysis and design of hierarchical web caching systems*, INFOCOM 2001. Proceedings IEEE , 2001, volume 3, 9p.

[20] CHEN, Xiangping, MOHAPATRA, P., *Lifetime behavior and its impact on Web caching*, IEEE Workshop on Internet Applications, 1999, 8p.

[21] CHENG, Kai, KAMBAYASHI, Y., *LRU-SP: a size-adjusted and popularity-aware LRU replacement algorithm for web caching*, Computer Software and Applications Conference, 2000. COMPSAC 2000, The 24th Annual International, 2000, 8p.

[22] CHIANG, Cho-Yu, UENO, M., LIU, M.T., MULLER, M.E., *Modeling Web caching schemes for performance studies*, Proceedings 2000 International Conference on Parallel Processing, 2000, 8p.

[23] CHIANG, Cho-Yu, LI, Yingjie, LIU, M.T., MULLER, M.E., *On request forwarding for dynamic Web caching hierarchies*, Proceedings 20th International Conference on Distributed Computing Systems, 2000 8p.

[24] CROVELLA, M. E., CARTER, R. L., *Dynamic server selection on Internet*, In Proceedings Of the Third IEEE workshop on the Architecture and Implementation of High Performance Comunication subsystems (HPCS'95), IEEE Comunication Society, Nova York, Agosto 1995, 5p.

[25] DAVISON, Brian D., *Online Web Caching Resources*. http://www.web-caching.com.

[26] DAVISON, Brian D., *A Web caching primer*, IEEE Internet Computing, Julho-Agosto, 2001, volume 5, 8p.

[27] DINGLE, A., *Web Cache Coherence*, Computer Networks and ISDN systems, 1996, volume 28, 14p.

[28] DYKES, S.G., JEFFERY, C.L., DAS, S., *Taxonomy and design analysis for distributed Web caching*, Systems Sciences 1999, HICSS-32 International Conference on Proceedings of the 32nd Annual Hawaii, 1999, 11p.

[29] DUCHAMP, Dan, *Prefetching Hyperlinks*, In Proceeding of the USENIX Symposium on Internet Technologies and Systems, 1999.

[30] DUVVURI, Venkata, SHENOY, Prashant, TEWARI, Renu, *Adaptative Leases: A strong Consistency Mechanism for the Worl Wide Web*, IEEE Infocom, 2000.

[31] FONSECA, Erik Luiz S., Hierarquias de Servidores Proxy Cache WWW: Instrumentação e Análise de Desempenho, Universidade Federal de Minas Gerais, Março, 1999.

[32] FOYGEL, D.; STRELOW, D., *Reducing Web latency with hierarchical cache-based prefetching*, Proceedings 2000 International Workshops on Parallel Processing, 2000, 6p.

[33] GADDE, Syam, *The Proxycizer Web proxy tool suite*, http://www.cs.duke.edu/ari/Proxycizer/.

[34] GRAY, C., CHERITON, D., *An Efficient Fault Tolerant Mechanism for Distributed File Cache Consistency*, Proceedings 12th ACM Symposium Operating Systems Principles, 1989, 9p.

[35] GWERTZMAN, J., SELTZER, M. *The Case for geographical push-caching*, In Proceedings Of the 1995 Workshop on Hot Operation Systems (HOTOS-V), 1995, 5p.

[36] GWERTZMAN, J., SELTZER, M., *Worl Wide Web Cache Consistency*, USENIX Annual Technical Conference, 1996, 10p.

[37] HOWARD, J. et al., *Scale and Performance in a Distributed File System*, ACM Trans. Computer Systems, Fevereiro 1998, volume 6, número 1, 31p.

[38] JIN, Jing-Wen, MEIRA, Silvio Lemos, SILVA, Fábio Q. B., *Making Web Client Caching Cooperate at LAN*, Journal of the Brazilian Computer Society, ISSN 0104-6500, Novembro 1998, volume 5, número 2, 11p.

[39] JONHSON, Tommy, *The WebJamma*, NRG, Computer Science Department, Virginia Tech White Paper, 1998. http://research.cs.vt.edu/nrg/webjamma.html.

[40] KAREDLA, R., LOVE, J.S., WHEERY, B. G., *Caching Strategies to Improve Disk System Performance*, IEEE Computer, Março, 1994, 7p.

[41] KAWAI, Eiji, CHINEN, Ken-ichi, YAMAGUCHI, Suguri, SUNAHARA, Hideaki, *An Analysis of the Number of ICP packets on the Distributed WWW Caching system*, International Workshops on Parallel Processing, Wakamatsu, Japan, Setembro, 1999.

[42] KENYON, C., *The evolution of Web-caching markets*, IEEE Computer, Novembro 2001, volume 34, 3p.

[43] KIM, Kyungbaek, PARK, Daeyeon, *Least Popularity-Per-Byte Replacement algorithm for a proxy cache Parallel and Distributed Systems 2001*, Proceedings. Eighth International Conference on ICPADS 2001, 2001, 8p.

[44] KOLETSOU, Mimika, VOELKER, Geoffrey M., *The Medusa Proxy – A tool for exploring User-perceived Web Performance*, Department of Computer Science and Engineering University of California, São Diego, 1997.

[45] KRISHANA, P., VITTER, Jeffrey, *Optimal Prediction for Prefetching in the Worst Case*, A shortened version appeared in Proceedings of Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, Janeiro, 1994, 10p.

[46] KRISHNAMURTHHY, B., WILLS, C.E., *Piggyback Server Invalidation for Proxy Cache Coherency*, Proceedings of the WWW-7 Conference, 1998, 10p.

[47] KRISHNAMURTHHY, B., WILLS, C.E., *Study of Piggyback Cache Validation for Proxy Caches in Word Wide Web*, Proceedings of the 1997 USENIX Symposium Internet Technology and Systems, Dezembro 1997, 13p.

[48] LAI, Guanpi, LIU, Mingkuan, WANG, Fei-Yue, ZENG, D., *Web caching: architectures and performance evaluation survey*, 2001 IEEE International Conference on Systems, Man, and Cybernetics, 2001, volume 5, 6p.

[49] LINDEMANN, Christoph, WALDHORST, Oliver P., *Analysis of Web Caching in the Gigabit Research Network G-Win*, University of Dortmund Department of Computer Science, Germany, Abril, 2001.

[50] LIU, Chengjie, CAO, Pei, *Maintaining Strong Cache Conssitency in the Worl Wide Web*, Proceedings of the 17th IEEE International Conference on Distribuded Computing Systems, Maio, 1997.

[51] LIU, Mingkuan; WANG, Fei-Yue; ZENG, D., YANG, Lizhi, *An overview of world wide web caching*, 2001 IEEE International Conference on Systems, Man, and Cybernetics, 2001, volume 5, 6p.

[52] MARKATOS, E. P., CHORNAKI, C. E., *Top-10 Approach to Prefetching the Web*, in Proceedings of the Eighth Annual Conference of the Internet Society (INET'98), Geneva, Switzerland, Julho, 1998.

[53] MEIRA, Wagner, FONSECA, Erik L. S., ALMEIDA, Virgílo A.F., MURTA, Cristina D., *Performance Analysis of WWW Cache Proxy Hierachies*, Journal of the Brazilian Computer Society, ISSN 0104-6500, Novembro 1998, volume 5, número 2, 15p.

[54] MEIRA, Wagner Jr., FONSECA, Erik L. S., MURTA, Cristina Duarte Murta, ALMEIDA, Vírgilio A F, *Analyzing Performance of Cache Server Hierarchies*, Proceedings of the XVIII International Conference of the Chilean Society of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1998, 8p.

[55] MENASCÉ, Daniel A., ALMEIDA, Virgílio A.F., DOWDY,Larry W., *Capacity Planning and Performance Modeling*, Primeira Edição, Nova Jersey: Editora Prentice Hall, 1994.

[56] MENAUD J.M., ISSAMY, V., BANATRE, M., *A scalable and efficient cooperative system for Web caches*, IEEE Concurrency, Julho-Setembro, 2000, volume 8, 7p.

[57] MICHEL, Scott, NGUYEN, Khoi, ROSENSTEIN, Adam, ZHANG, Lixia, *Adaptative Web Caching: Towards a new caching architecture*, In Third International Caching Workshop, Junho 1998.

[58] MICROSOFT, Microsoft Proxy Server. http://www.microsoft.com/isaserver/evaluation/previousversions/default.asp

[59] MOGUL, J., *Network behavior of a busy web server and its clients*, Technical Report Research Report 95/5, Digital Equipment Corporation, Outubro 1995.

[60] MOSBERGER, D., JIN, T., *httperf: A tool for measuring web server performance*, in WISP, ACM, Madison, WI, Junho 1998, 9p.

[61] MURTA, Cristina D., ALMEIDA, Vírgilio A.F., *Characterizing Response Time of WWW Caching Proxy Servers*, Workshop on Workload Characterization, realizado em conjunto com o IEEE/ACM Micro'31, International Symposium on Microarchitecture, Dallas-Texas, 1998, 7p.

[62] MURTA, Cristina Duarte. *Modelo de Particionamento de Espaço para Cache da World Wide Web*, Tese Doutorado, Universidade Federal de Minas Gerais, 1999.

[63] MURTA, Cristina Duarte, ALMEIDA, Virgílio A.F., *Using performance maps to understand the behavior of Web caching policies*, Proceedings The Second IEEE Workshop on Internet Applications, 2001, 7p.

[64] MURTA, Cristina D., ALMEIDA, Virgílio A. F., MEIRA, Wagner Jr., *Efficient Storage Management in World Wide Web Caches*, Proceedings of the ITC'16 – 16th International Telegraffic Congress, Elsevier Science,

Edinburgh International Conference Centre, p. 1189-1198, UK, 7-11 June 1999.

[65] NELSON, M., WELCH, B., OUSTERHOUT, J., *Caching in the Sprite Network File System*, ACM Trans. Computer Systems, Fevereiro, 1998, volume 6, número1.

[66] NLANR (National Laboratory for Applied Network Research), *A distributed testbed for national information provisioning*, http://ircache.nlanr.net/.

[67] O'NEIL, E. J., O'NEIL, P. E., WEIKUM, G., *The LRU-K Page Replacement Algorithm for Database Disk Buffering*, Proceedings of ACM SIGMOD International Conference on Management of Data, New York, 1993, 10p.

[68] PATTERSON, David A., HENESSY, John L., *Computer Architecture a Quantitative Approach*, Segunda Edição, Califórnia: Editora Morgan Kaufmam Publishers Inc. San Francisco, 1996.

[69] PBS, *Life on the Internet*, http://www.pbs.org/internet/timeline/

[70] POVEY, D., HARRISON, J., *A distributed Internet Cache*, in Proc. 20th Australian Computer Science Conf., Sydney, Australia, Fevereiro 1997.

[71] RAJ, Jain, The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling, Wiley Professional Computing, 1992.

[72] RIZZO, L., VICIANO, L., *Replacement Policies for a Proxy Cache*, Technical report rn/98/13, University College London, Department of Computer Science, UK, 1998.

[73] RNP, Rede Minas. http://www.redeminas.br/estatisticas/cache/, http://www.redeminas.br

[74] ROBINSON, J. T., DEVARAKONDA, M. V., *Data Cache Management Using Frequency-based Replacement*, Performance Evaluation Review 18(1), Maio 1990, 9p.

[75] RODRIGUEZ, Pablo, SPANNER, C., BIERSACK, E.W., *Analysis of Web caching architectures: hierarchical and distributed caching*, IEEE/ACM Transactions on Networking, Agosto, 2001, volume 9, 15p.

[76] ROSS, K., *Hash Routing for Collections of Shared Web Caches*, IEEE Network Magazine, Novembro, 1997, 37p.

[77] ROUSSKOV, Alex, SOLOVIEV, Valey, *On Performance of Caching Proxies*, Conferência ACM SIGMETRICS'98, Junho 1998.

[78] ROUSSKOV, Alex, SOLOVIEV, Valey, *A Performance Study of the Squid Proxy on HTTP/1.0*, World Wide Web Journal, WWW Characterization and Performance Evaluation, 1999.

[79] ROUSSKOV Alex, WESSELS, Duane, WEB POLYGRAPH, Proxy Performance benchmark, http://polygraph.ircache.net/

[80] SANTANA, Regina H. C. et al., SANTANA, Marcos J. et al., ORLANDI, Regina C. et al., *Técnicas para avaliação de desempenho de sistemas computacionais*, Notas Didáticas, Instituto de Ciências Matemática de São Carlos, USP, Setembro de 1994.

[81] SHARP, Edward, *Caching Performance Realities*, www.web-caching.com, Janeiro, 2001.

[82] SCHAWARTATZ, M.F. et al., CHANKHUNTHOD, A. et al., DANZIG, P.B. et al., NEERDAELS, C. et al., *"Hierarchical Internet Object Cache"*, in

Proceedings 1996 USENIX Technical Conference, San Diego, CA, Janeiro 1996.

[83] SCHUBERT, Olaf, *Squid Log Analyzer*, http://squidlog.sourceforge.net/

[84] SELTZER, M., *The World Wide Web: Issues and challengers*, Division of Engineering and Applied Sciences, Harvard University, Presented at IBM Almaden, Julho, 1996.

[85] SELVAKUMAR, S., PRABHAKAR, P., *Implementation and comparison of distributed caching schemes*, Proceedings of IEEE International Conference on Networks 2000 (ICON 2000), 2000, 9p.

[86] Soares, L. F. G., *Modelagem e Simulação Discreta de Sistemas*, Editora Campus Ltda, 1992

[87] TEWARI, Renu, DAHLIN, Michael, VIN, Harrick M., KAY, Jonathan S., *Beyond Hierarchies: Design Considerations for distributed caching on the Internet*, in Proceedings ICDCS '99 Conf., Austin, TX, Maio, 1999.

[88] TEWARI, Renu, DAHLIN, Michael, VIN, Harrick M., KAY, Jonathan S., *Design Considerations for Distributed Caching on the Internet*, IBM T.J. Watson Research Center Department of Computer Sciences Cephalapod Proliferationists, 1999.

[89] TRANQUILI, Nico, *Squid Times*, http://www.cineca.it/~nico/squidtimes.html

[90] UNINETT, *Project Desire Webcache*, UNINETT, http://www.uninett.no/prosjekt/desire/

[91] VÖCKLER, Jens-S., *Http Blaster*, http://www.cache.dfn.de/DFN-Cache/Development/blast.html.

[92] VOELKER, G. M., ANDERSON, E. J., *Implementing Cooperative Prefetching and Caching in a Globally-Managed Memory System*, in Proceedings of the 1998 ACM SIGMETRICS Conference on Performance Measurement, Modeling and Evaluation, Junho, 1998.

[93] WEBPACITY, *Webpacity*, http://webpacity.com

[94] WESSELS, D., CLAFFY K., *Application of Internet Cache Protocol (ICP)*, version 2. Networking Group, RFC 2187, Setembro 1997.

[95] WESSELS, D., CLAFFY K., *Internet Cache Protocol (ICP)*, version 2. Networking Group, RFC 2186, Setembro 1997.

[96] WESSELS, Duane, Information Resource Caching FAQ, http://ircache.nlanr.net/Cache/FAQ/ircache-faq.html.

[97] WESSELS, Duane, ROUSSKOV, Alex, CHISHOLM, Glenn, *SQUID*, http://www.squid-cache.org/

[98] WONG, Kin Yeung, YEUNG, Kai Hau, *Site-based approach to Web cache design*, IEEE Internet Computing, Setembro à Outubro 2001, 7p.

[99] YING, Jian et al., ALVISI, Lorenzo et al., *Colume Leases for Consistency in Large-Scale Systems*, IEEE Transactions on Knowledge Data Engineering, Julho 1999, volume 11, número 4.