UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

COORDENAÇÃO DO CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ALEXANDRE HEKERMANN HILBERT BUSS

GUSTAVO KULIK SILVA

UTILIZANDO *MEMCACHED* PARA DIMINUIR O TEMPO DE RESPOSTA DE APLICAÇÕES WEB

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2013

ALEXANDRE HEKERMANN HILBERT BUSS

GUSTAVO KULIK SILVA

UTILIZANDO *MEMCACHED* PARA DIMINUIR O TEMPO DE RESPOSTA DE APLICAÇÕES WEB

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, da COADS, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Willian Massami Watanabe

Co-orientador: Prof. Wellton Costa de Oliveira

PONTA GROSSA

2013

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa
Nome da Diretoria
Nome da Coordenação
Nome do Curso

TERMO DE APROVAÇÃO

UTILIZANDO MEMCACHED PARA DIMINUIR O TEMPO DE RESPOSTA DE APLICAÇÕES WEB

por

ALEXANDRE HEKERMANN HILBERT BUSS
GUSTAVO KULIK SILVA

Este(a) Trabalho de Conclusão de Curso (TCC) foi apresentado(a) em 26 de março de 2013 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Willian Massami Watanabe
Prof.(a) Orientador(a)

_____
Wellton Costa de Oliveira
Membro titular

_____
Thalita Scharr Rodrigues
Membro titular

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

Dedicamos este trabalho a todos os nossos familiares e amigos que nos apoiaram nessa longa caminhada. Obrigado pela compreensão.

AGRADECIMENTOS

Primeiramente gostaríamos de agradecer a Deus por toda força e sabedoria enviada para a realização deste trabalho. Toda a energia positiva para enfrentar os obstáculos para no final poder dizer com certeza: Venci, valeu a pena!

Aos nossos pais Caroline e Edmilson, Reinoldo e Bernadete, por nos apoiarem durante todo o tempo da graduação e fazer possível para termos chegado até aqui.

A Universidade Tecnológica Federal do Paraná (Campus Ponta Grossa) que nos apoiou e contribuiu ao nosso crescimento profissional.

A todos os amigos que ajudaram de certa forma e foram compreensivos em nossa ausência.

Aos professores Willian M. Watanabe e Wellton Costa de Oliveira, pela grande ajuda e participação em nossa formação acadêmica e pessoal.

Muito Obrigado!

RESUMO

HILBERT, Alexandre, KULIK, Gustavo. **Utilizando *Memcached* para diminuir o tempo de resposta de aplicações web.** 2013. 32. Trabalho de Conclusão de Curso - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2013.

Este trabalho é um estudo de métricas de performance (métodos), efetuadas para medir e analisar a diferença em velocidade de respostas a requisições HTTP – *Hypertext Transport Protocol*. Existem diversas métricas de performance, como por exemplo a taxa de crescimento de usuários ativos, taxa de crescimento de faturamento, como no caso do *eCommerce*, entre outras. Neste trabalho, serão abordadas métricas de velocidade de resposta a requisições web, ou seja, o tempo que a aplicação demora para responder as requisições feitas pelo usuário, com enfoque no gargalo entre a aplicação (*task manager* online desenvolvido utilizando o CMS – *Content Management System* - Drupal) e banco de dados (desenvolvido em MySql). O método estudado para melhorar a performance do site web é o uso de *Memcached*, que realiza o carregamento dos dados na memória para serem apresentados com mais rapidez ao usuário. Para medir a velocidade de resposta a estas requisições é realizado o uso do software *JMeter*.

Palavras-chave: Task manager, Drupal, *Memcached*, Requisições HTTP e *JMeter*.

ABSTRACT

HILBERT, Alexandre, KULIK, Gustavo. ***Memcached* using to reduce the response time for web applications**. 2013. 32. Completion of course work - Federal Technological University of Paraná. Ponta Grossa, 2013.

This work is a study of performance metrics, made to measure and analyze the difference in response time of HTTP requests - Hypertext Transport Protocol. There are several performance metrics, such as the growth rate of active users, the growth rate of billing, such as eCommerce, among others. However this work focuses on HTTP requests, and the time that the application takes to respond to requests from the user perspective, focusing on the bottleneck between the application (online task manager developed using CMS - Content Management System - Drupal) and database (developed in MySql). The method we used to improve the performance of the web site is the *Memcached*, which implements a dynamic memory *cache* strategy for every database access. This work reports an experiment that measured the response time of applications that use *Memcached* and application that do not use *Memcached*, using the *JMeter* tool.

Keywords: Task manager, Drupal, *Memcached*, HTTP Requisitions, *JMeter*

SUMÁRIO

1. INTRODUÇÃO

O mercado cada vez mais pressiona os departamentos de TI - Tecnologia da Informação - para serem capazes de processar grandes volumes de dados em tempo hábil e de forma correta. O investimento em tecnologia da informação chega a mais da metade dos gastos anuais de capital na maioria das empresas no setor de serviços. A velocidade com que essa tecnologia e estes investimentos vêm sendo demandados, põe a prova não só o hardware, como também toda a infraestrutura de software.

Um sistema de informação pode ser definido como um conjunto de componentes inter-relacionados trabalhando juntos para coletar, recuperar, processar, armazenar e distribuir informações com a finalidade de facilitar o planejamento, o controle, a coordenação, a análise e o processo decisório em organizações (LAUDON e LAUDON, 1999).

Este processo é uma tarefa árdua que demanda enorme esforço tanto dos desenvolvedores para produzir código, quanto dos administradores de rede, para realizar otimização nos diversos componentes de um único serviço.

Como o desempenho de aplicações é algo significativo atualmente, deve-se focar principalmente nos gargalos de requisições. Ou seja, gargalos são onde o fluxo de dados é maior e devem ser tratados corretamente, diminuindo os custos operacionais, aumentando o desempenho e a dinâmica das aplicações.

Para consolidar esta ideia, se faz necessário o uso de novas tecnologias de otimização como, por exemplo, o *Memcached*. Além disso, o uso de técnicas de medição também é necessário para comprovar que o uso das novas tecnologias são realmente viáveis para as aplicações. Essas técnicas de medições podem ser chamadas também de teste de desempenho.

O *teste de desempenho* é usado para descobrir problemas de desempenho que podem resultar da falta de recursos do lado do servidor. Exemplos são a largura de banda inadequada, recursos de banco inadequados, recursos deficientes do sistema operacional, podendo causar degradação de desempenho cliente-servidor. O teste visa descobrir como o sistema vai se portar diante de um elevado número de usuários, transações e grande volume geral de dados. E visa também verificar possíveis modificações para melhorar o desempenho (PRESSMAN, 2011).

> “Qualquer recurso do sistema, como hardware, software ou largura de banda, que estipule limites definidos no fluxo de dados ou na velocidade de processamento representa um gargalo. Em aplicações Web, os gargalos afetam diretamente o desempenho e a escalabilidade limitando a quantidade de *throughput (taxa de transferência)* de dados ou restringindo o número de conexões da aplicação. Esses problemas ocorrem em todos os níveis da arquitetura do sistema, incluindo a camada de rede, o servidor Web, o servidor de aplicações e o servidor de banco de dados.”
> (ORACLE, 2009, p.3)

O foco deste trabalho é apresentar uma alternativa para amenizar o gargalo de aplicações web que ocorrem no banco de dados. Para isso foi utilizado o *Memcached*, que é um método que carrega na memória secundária os dados da primeira requisição. A partir da segunda requisição estes dados já são enviados mais rapidamente ao cliente, pois estão carregados em memória, sem a necessidade de consulta nos dispositivos de entrada e saída, como o disco rígido do servidor.

Este trabalho apresenta a seguinte estrutura: no próximo capítulo é apresentado o embasamento teórico, seguido pelo desenvolvimento, resultados e conclusões.

2. EMBASAMENTO TEÓRICO

Neste capítulo serão abordadas as técnicas utilizadas para o desenvolvimento e aperfeiçoamento de uma aplicação web, desde a fundamentação e desenvolvimento de um web site até a parte das métricas de performance de requisições web. Feito isso, pode-se visualizar a diferença de performance entre aplicações web que utilizam um mecanismo de *cache* em memória (*Memcached*) e aplicações web que não o utilizam.

2.1. WEB

A *World Wide Web* - www - é o segmento da Internet de crescimento mais acelerado. Suas características de interface gráfica e de hipertexto vêm fascinando os usuários da Internet e a mídia, de uma forma que nenhum outro conjunto de utilitários da Internet conseguiu. As empresas comerciais, escolas, setores do governo, organizações sem fins lucrativos e mesmo indivíduos isolados estão recorrendo à *Web* para promover a si mesmos e aos seus produtos diante de uma audiência que se espalha por todos os países. Devido à popularidade e à eficácia como recurso de marketing, a *World Wide Web* está a ponto de se transformar no estabelecimento comercial eletrônico da década (CHANDLER; KRIKNER; MINATEL, 1996).

A idéia de web site constantemente remete aos antigos sites, que eram lentos e trabalhosos, dificultando a utilização do usuário. Pode-se deparar com sites em que as requisições demorem a serem respondidas, ou travam e não atendem as necessidades do usuário. Também ocorre de o usuário requerer uma informação e acaba recebendo outra diferente da esperada.

Com base nisso, a *Web* nos traz diversos problemas a serem solucionados, e um deles é o desempenho de requisições HTTP. O usuário espera uma alta performance e que o serviço atenda as expectativas e funções designadas. O propósito deste trabalho é desenvolver uma aplicação web que gere resultados mais rapidamente ao usuário, atendendo as necessidades do mesmo com praticidade e coerência.

2.2. CMS

Um CMS - *Content Management System*, ou Sistema de Gerenciamento de Conteúdo como o próprio nome já diz, é um software capaz de publicar, editar e modificar determinados conteúdos definidos pelo usuário. Um CMS possui uma interface própria de gerência de conteúdo, podendo ser alterada de acordo com as necessidades do usuário.

Em outras palavras, um CMS é uma ferramenta que permite a um editor criar, classificar e publicar qualquer tipo de informação em uma página web. Geralmente, os CMS's trabalham contra um banco de dados, de modo que o editor simplesmente atualiza um banco de dados, incluindo nova informação ou editando informações existentes (ALVAREZ, 2013).

O primeiro CMS criado foi anunciado no final da década de 90, o objetivo principal da sua criação era facilitar à complexa e trabalhosa tarefa de desenvolvimentos web. A repercussão foi significativa e os CMS's se tornaram uma ferramenta popular entre os desenvolvedores de blogs, o que acabou inundando a web com milhares de sites próprios e diversos.

Existem diversos tipos de CMS's, os mais populares são utilizados para criar blogs, como por exemplo, o WordPress e o Joomla. Existem também CMS's com maior afinidade para a criação de fóruns, comércio eletrônico, como por exemplo, o PrestaShop e o OsCommerce, Wikis, sistemas de ensino e comercial, como por exemplo o Vignette. Pode- se observar no gráfico abaixo a utilição de alguns CMS`s mais populares.

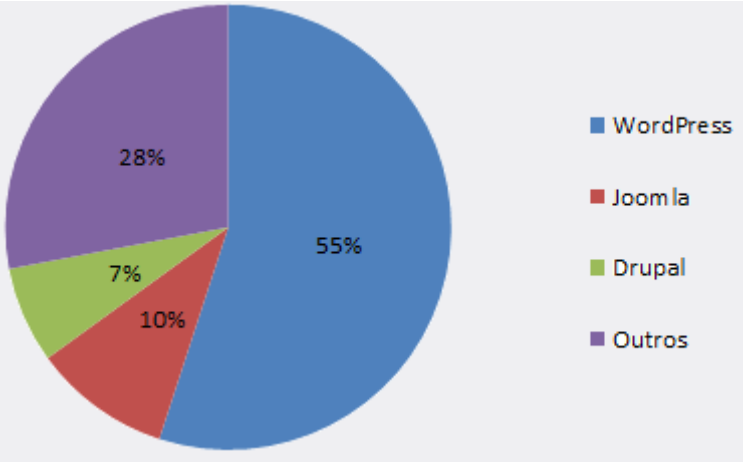

Figura 1 – Gáfico utiizacão dos CMS`s.

Fonte: www.yogh.com.br/blog/cms/por-que-escolher-o-wordpress/ **(2013)**

O principal conceito de CMS's está relacionado à questão do reuso, que, pode ser entendido como uso de conceitos ou produtos previamente adquiridos ou construídos em uma nova situação (BIGGERSTAFF AND PERLIS, 1989).

O reuso nasceu da observação de que os softwares frequentemente seguem padrões similares, sendo possível explorá-los e assim, empregar menos esforços em solução para problemas já resolvidos anteriormente(MEYER, 2000).

O reuso não é somente aplicável a fragmentos de código fonte, mas é uma forma eficaz de aproveitar os conhecimentos sobre o desenvolvimento de sistemas (DA SILVA SANTOS, 2004).

Entre outras palavras, no contexto deste trabalho, o reuso se dá pela forma como os CMS's são produzidos, ou seja, os módulos e atributos são padrões, podendo haver o reuso, não havendo necessidade de se criar a mesma funcionalidade diversas vezes, sempre que for necessária a sua utilização, este é um dos pontos chaves de um CMS.

2.2.1. Características

A função principal de um CMS é exibir as informações em sites web, entretanto esses sistemas variam muito conforme a sua especialidade. Os CMS's mais robustos, frequentemente possuem um custo elevado para o cliente. Porém, garantem um conjunto de características específicas para o desenvolvimento, como por exemplo, indexação, busca e recuperação de informações. Esses CMS's mais complexos podem ser chamados e sistemas corporativos.

Um CMS pode servir também para armazenamento central, seja de vídeos, filmes, imagens, dados científicos etc. como também para revisão, controle de publicações e documentação.

2.3. DRUPAL

Neste trabalho foi utilizado o CMS Drupal. O Drupal nada mais é que um CMS próprio para desenvolvimento de web sites. Ele foi lançado no ano de 2000 e implementado em PHP[1] – *Hypertext PreProcessor*. O Drupal possui o código-fonte

[1]http://www.php.net/

livre, ou seja, qualquer pessoa pode obter diretamente do site e realizar alterações e modificações no seu código fonte sem problema algum.

2.3.1. Características de Usabilidade

O Drupal possui uma tela de programação fácil de ser compreendida e utilizada. Seus botões para adição e remoção de campos são simples e garantem que um usuário sem conhecimento em programação possa dominar o assunto e construir um web site próprio sem muito esforço. Para esse efeito o Drupal foi dividido em módulos, ou seja, o usuário pode escolher uma "cara" nova para o seu site quando for realizar a confecção do mesmo. Estes módulos são facilmente obtidos do próprio web site do Drupal, onde o usuário pode visualizar os módulos que melhor se encaixam com a necessidade do problema. Existem, por exemplo, módulos para e-commerce, blogs, sites de notícias, etc.

Estes módulos podem disponibilizar "*templates*", ou seja modelos pré-definidos de um web site. Por exemplo, um *eCommerce*, depois de efetuado o download do módulo *eCommerce* no site do Drupal, é possível também escolher qual o melhor template para se trabalhar com este tipo de *eCommerce*, de acordo com a figura 2 que ilustra o Retail Shop.

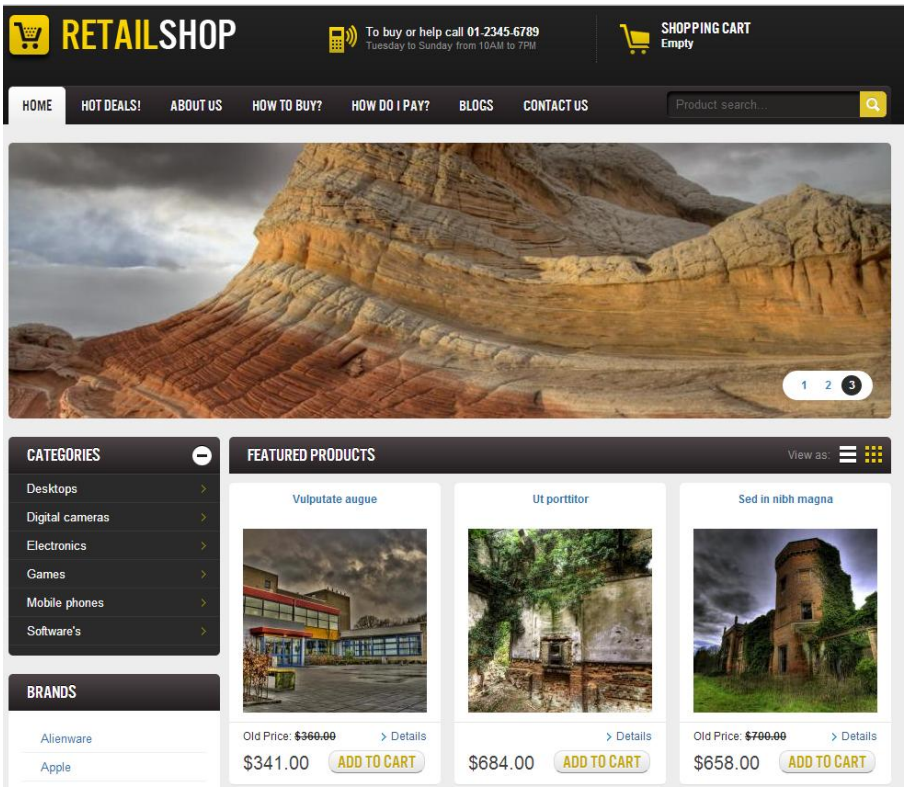

Figura 2 - Modelo de Template para o módulo *eCommerce*.

Fonte: RetailShop.com (2013.)

2.3.2. Instalação

O Drupal pode ser considerado de fácil instalação, necessitando apenas ser criada uma base de dados no banco e o próprio Drupal irá criar todas as tabelas e atribuições necessárias para o uso do mesmo.

Depois de efetuado o *download*, o Drupal possui uma tela de *feedback* que auxilia usuário na instalação. Se ocorrem erros, informam-se onde estam os erros, e geralmente, até como corrigi-lo.

No Drupal, a maior parte de funcionalidades funciona a partir de módulos, que são arquivos PHP onde há funcionalidades personalizadas que se integram junto ao Drupal. Na própria página do CMS, há uma lista de módulos com sua descrição, instruções e a sua possibilidade de *download*. Para utilizar o módulo que foi adquirido da página web do Drupal, é necessário somente que seja extraído na pasta de destino dentro da instalação do Drupal ou pela própria página de upload de módulos, e depois ativá-lo no painel de administração do site. São dois tipos de módulos encontrados na página do Drupal: os módulos de núcleo (*core modules*) que estão incluídos no Drupal e os módulos de contribuição (*Add-On Modules*). Eles

são oferecidos pela comunidade Drupal e podem ser adquiridos e habilitados separadamente (DRUPAL PARÁ, 2013).

A figura 3 apresenta a aba destinada a instalação de módulos opcionais no CMS, além disse eles poder ser instalado porem desativados de acordo com a necessidade do usuário.

Figura 3 - Sessão de módulos do Drupal.
Fonte: Painel Drupal (2013).

2.4. BANCO DE DADOS

Segundo Korth**,** um banco de dados “é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico”, ou seja, sempre que for possível agrupar informações que se relacionam e tratam de um mesmo assunto, pode-se dizer que é um banco de dados (Korth, 1994).

O banco de dados é essencial para uma aplicação web. Nesse trabalho o qual foi utilizado foram inseridos diversos tipos de informações para preencher a base de dados.

O CMS Drupal, trabalha com uma base de dados feita em MySql, na qual o usuário cadastra os dados a serem tratados pelo administrador do site. O ponto

importante trabalhado neste projeto é o gargalo que essas aplicações inferem sobre a performance e a dinâmica da aplicação.

Quando um usuário faz uma requisição ao site, ele deve buscar as informações na base de dados e isto demorar. Em um site popular, o número de usuários acessando e fazendo requisições são simultâneas, e conforme o tamanho do banco de dados e das tuplas – linhas de informações – nele contidas, a busca pela informação correta demora, atrasando o resultado final ao usuário.

Para reduzir o tempo de busca destas informações, chamados de gargalos de aplicação, se faz necessário o uso de técnicas de aperfeiçoamento de requisições, como por exemplo o *Memcached*, abordado no próximo tópico.

2.5. *CACHE* E *MEMCACHED*

2.5.1. Memória *Cache*

Na computação, *cache* é um dispositivo de acesso rápido, interno a um sistema, que serve de intermediário entre um operador de um processo e o dispositivo de armazenamento ao qual esse operador acede. A vantagem principal na utilização de um *cache* consiste em evitar o acesso ao dispositivo de armazenamento, que pode ser demorado, armazenando os dados em meios de acesso mais rápido.

O uso de memórias *cache* visa obter uma velocidade de acesso a memória próxima da velocidade de memórias mais rápidas, e ao mesmo tempo disponibilizar no sistema uma memória de grande capacidade, a um custo similar de memórias semicondutoras mais baratas.

Com os avanços tecnológicos vários tipos de *cache* foram desenvolvidos. Atualmente há *cache* em processadores, disco rígidos, sistemas, servidores, nas placas-mãe, clusters de banco de dados, entre outros. Qualquer dispositivo que requeira do usuário uma solicitação/requisição a algum outro recurso, seja de rede ou local, interno ou externo a essa rede, pode requerer ou possuir de fábrica o recurso de *cache*.

Dentro deste contexto será abordado um tipo específico de *cache*, denominado *Memcached*, o qual será visto no próximo tópico.

2.5.2. *Memcached*

Os usuários podem frustar-se se a aplicação web levar muitos minutos para carregar o conteúdo, enquanto seus concorrentes carregam um conteúdo similar em segundos. Nada é mais desconcertante que uma página web que responde instantaneamente em algumas situações e depois parece entrar em um estado de espera infinita em outras. Essas situações ocorrem frequentemente na internet e estão relacionadas com desempenho (PRESSMAN, 2011).

Ao entrar em algum site o usuário deseja que suas necessidades sejam atendidas rapidamente e com eficácia. Problemas ocorrem neste quesito, pois os usuários efetuam requisições e elas demoram a ser respondidas, ou até mesmo requisições não são respondidas, travando o site e fazendo com que o usuário recomece tudo novamente.

Esse problema era mais comum antigamente, todavia mesmo hoje em dia, quando os acessos se tornam numerosos, alguns servidores não suportavam o número de requisições. Conseqüentemente, cessavam suas operações de resposta, negando o acesso ao web site pelos seus usuários[2].

O problema sobre o gargalo existente entre a aplicação e o banco de dados foi parcialmente resolvido criando-se um *cache*, ou seja, algumas requisições eram tratadas sem o uso do disco rígido, mais especificamente na memória. Com isso o processo era mais rápido e menos suscetível a erros e quedas de sistema.

Porém, a popularização da internet tornou até mesmo este processo de *cache* obsoleto. Considerando isso, em 2003, Brad Fitzpatrick começou a desenvolver o chamado *Memcached*, naquela época o objetivo era melhorar o desempenho do site LiveJournal.

[2] No Enem 2012 houve uma sobrecarga no sistema do MEC devido ao grande numero de acessos simultâneos, isto fez com que o sistema ficasse fora do ar por meia hora, até que o servidor fosse restabelecido, deixando os candidatos impossibilitados de ver sua pontuação no exame.

2.5.3. Características

O *Memcached* é um projeto de software livre idealizado para fazer uso da memória dinâmica sobressalente em muitos servidores para agir como um *cache* de memória para informações acessadas com freqüência. O elemento chave é o uso da palavra *cache:* o *Memcached* fornece armazenamento temporário, em memória, das informações que podem ser carregadas de outro local (Brown, 2010).

O *Memcached* possui uma arquitetura do tipo cliente-servidor. O servidor mantém um vetor associativo chave-valor, o cliente completa esse vetor e faz a consulta. No *cache* simples, apenas algumas requisições eram atreladas à memória, já no *Memcached*, praticamente tudo que será usado do disco rígido será armazenado na memória dinâmica. Com isso, diminui-se em grande parte a incidência de requisições sem resposta rápida, uma vez que a busca não demora a ser encontrada, pois já está carregada na memória, a Figura 4 mostra basicamente como funciona o *Memcached* a partir do diagrama de fluxo.

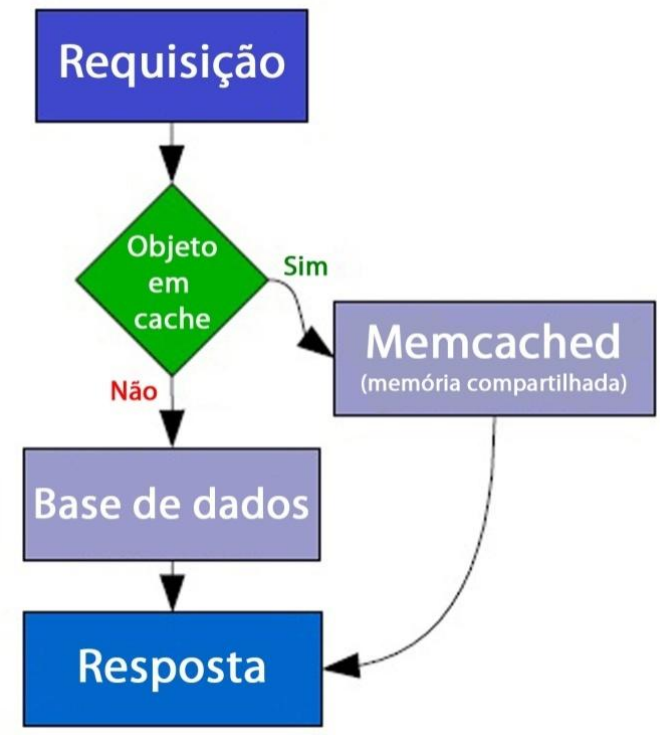

Figura 4 - Diagrama de funcionamento do *Memcached*.

Fonte: http://cerberusweb.com/book/latest/_images/*Memcached*_flowchart.png (2013).

Entretanto, existe um problema: a lotação da memória que pode sobrecarregar o sistema. Para esse problema ser resolvido, os vetores possuem um

valor de tempo em que ficam carregados na memória. Ou seja, quando um usuário faz uma requisição e ela não está na memória, ela é carregada e permanece lá por um tempo. Se nenhum outro usuário refizer a mesma requisição dentro de um determinado período de tempo, ela é descartada novamente ao disco. Já os vetores que possuem grande número de acessos, nunca são jogados de volta para o disco, permanecem sempre na memória. Desse modo, o *Memcached* garante uma velocidade de execução limpa e rápida, retornando as respostas ao usuário com agilidade e consistência.

2.5.4. *Cache* Distribuído

O servidor do *Memcached* é somente um *cache* armazenando valores com relação a chaves em uma rede. Se houver a possibilidade de utilizar várias máquinas como servidores de *Memcached*, pode ser definida uma instância do *Memcached* em cada máquina, para fornecer uma grande quantidade de armazenamento de *cache* em RAM na rede (Brown, 2010).

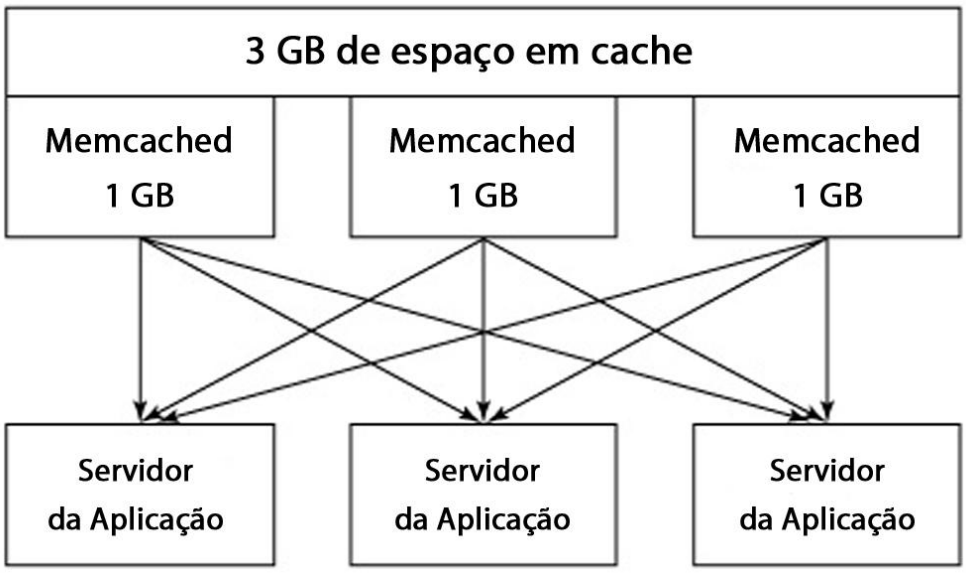

Figura 5 - O funcionamento de *Memcached* distribuído.
Fonte: IBM Corporation (2013).

Com base na figura 5 pode-se perceber que o *Memcached* distribuído não é usado em apenas um servidor, ele é distribuído entre 2 ou mais servidores. Assim, pode-se separar o *cache* entre várias máquinas em rede, ganhando maior tamanho de memória e armazenamento de *cache*.

O problema dessa aplicação é como tratar, qual servidor irá armazenar seu par chave/valor e como determinar qual servidor deverá acessar para recuperar este dado.

Em vez de ter que determinar estas informações, os clientes do *Memcached* usam um algoritmo de *hashing* simples na chave que for especificada ao armazenar a informação. Ao armazenar ou obter informações de uma lista de servidores do *Memcached*, o cliente deriva um valor numérico da chave usando um algoritmo de *hashing* consistente. Para exemplificar, a chave *mykey* é convertida para o valor 23875. Não importa se o usuário está armazenando ou obtendo informações, sempre usará a mesma chave como identificador exclusivo para carregar do servidor do *Memcached*, portanto "*mykey*" sempre obterá o *hash* com um valor de 23875 (Brown, 2010).

2.6. *JMETER*

O *JMeter* é uma ferramenta de teste de carga, performance e stress. Comumente é utilizada para realizar testes em aplicações web, que constituem o principal foco deste trabalho.

O Apache *JMeter* pode ser usado para testar o desempenho tanto em recursos estáticos e dinâmicos. Ele pode ser usado para simular uma carga pesada em um servidor de rede, ou objeto para testar a sua força ou para analisar o desempenho global no âmbito de diferentes tipos de carga (Foundation, 2013).

Para os testes realizados a partir do *JMeter*, é necessário criar um plano de testes incluindo os elementos a serem testados. Esses elementos podem ser: *Thread Group, Controllers, Listeners, Timers, AssertionsConfigurationElements, Pre-Processor Elements e Post-Processor Elements.* No caso deste trabalho será utilizado o *Post-Processor Elements,* que executa uma ação depois de fazer a requisição, usado para mostrar as respostas da requisição.

2.6.1. Funcionamento

O primeiro conceito básico utilizando é o de usuários, que no *JMeter* são representados por *threads* – tópicos. Dentro de um script as threads executarão a seqüência de passos determinada, onde cada thread simula um usuário fazendo uma requisição HTTP ao web site. Cada passo executado pelo “usuário” é denominado *sampler* – amostrador – que segue uma seqüência lógica de utilização da aplicação. O passo seguinte é coletar as informações fornecidas pelo software com o retorno dos *samplers*, verificando se o conteúdo está de acordo com o esperado e se o tempo de resposta é aceitável, este é o papel de um *listener* – ouvinte.

Assim, podem ser gerados dados e tabelas comparativas, mostrando a melhoria de desempenho na utilização de *Memcached* para aperfeiçoar os recursos web.

3. DESENVOLVIMENTO

Para a construção, configuração do sistema e testes de performance foram utilizadas algumas técnicas e ferramentas indispensáveis para o funcionamento do sistema, que serão descritas a seguir:

3.1. Instalação do servidor apache;
3.2. Instalação do banco de dados MySql;
3.3. Instalação do interpretador de linguagem PHP;
3.4. Instalação do gerenciador de banco de dados *PhpMyAdmin*;
3.5. Instalação do Drupal;
3.6. Instalação do *Memcached*;
3.7. Configuração do *Memcached*;
3.8. Configuração do *Memcached* para utilização com Drupal.

Primeiramente em uma estação com a configuração de hardware utilizando um processador AMD athlon x2 2.9 ghz e 6 gb de memória RAM com velocidade de 800mhz e disco rígido sata II, e instalado a distribuição Linux Ubuntu Server 12.04 64 bits, foram instalados os seguintes programas: Apache, Php, MySql e *PhpMyAdmin*, para garantir as características de um servidor web a estação.

3.1. INSTALAÇÃO DO SERVIDOR APACHE

Para instalar esses programas é indispensável a permissão de root (usuário administrador) entao, antes do procedimento foi efetuado o login como super usuário, no terminal de comandos do Linux.

Começando com o Apache, no terminal foi utilizado o APT (AdvancedPackaging Tool - em português Ferramenta de Empacotamento Avançada) que é um gerenciador de pacotes para o Sistema Operacional Linux.

3.2. INSTALAÇÃO DO BANCO DE DADOS MYSQL

Para instalação do banco de dados Mysql foram necessários dois pacotes: o mysql-server, que é o servidor em si, e o mysql-client que faz a comunicação com o

banco. Na instalação do banco é necessário criar uma senha de acesso ao usuário administrador do banco de dados, como mostrado na figura 6:

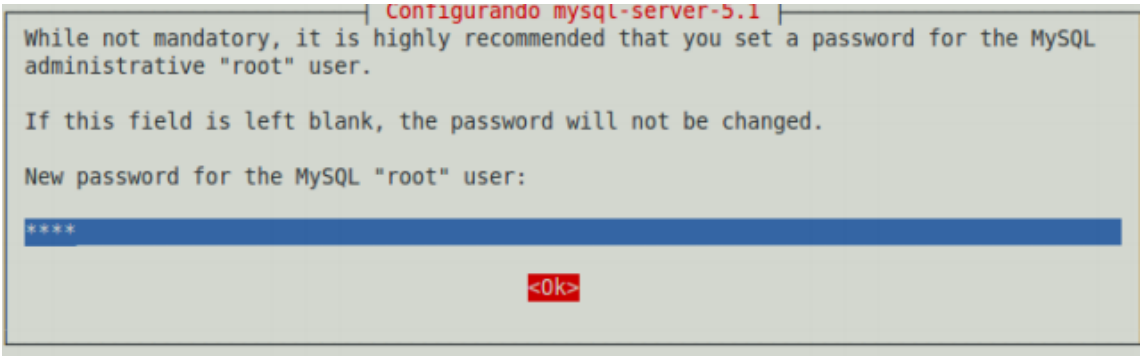

Figura 6 - Definindo uma senha para o usuário administrador do banco de dados
Fonte:Configuração MySql(2013).

Essa senha é importante, pois será utilizada posteriormente.

3.3. INSTALAÇÃO DO INTERPRETADOR DE LINGUAGEM PHP

Depois das atividades 1 e 2, para instalar o php foram necessários os programas:

php5 - php em si, libapache2-mod-php5 - pacote que fornece o modulo php5 para o servidor Apache2, e alguns pacotes complementares como:

php5-mysql - pacote que fornece módulos para conexões ao banco de dados MySql diretamente a partir de scripts Php,

php5-curl - é uma biblioteca para obter arquivos de FTP (File TransferProtocol – Protocolo de Transferência de Dados) e HTTP,

php5-gd – módulo para manipular gráficos diretamente a partir de scripts Php

php5-idn - módulo de internacionalização de caracteres no Php5,

php5-imagick –pacote fornece um invólucro para biblioteca ImageMagick,

php5-imap – módulo para funções Imap em scripts Php,

php5-mcrypt - módulo para funções Mcrypt em scripts Php,

php5-mhash - módulo para funções Mhash em scripts Php,

php5-ming - módulo para funções Ming em scripts Php,

php5-ps – extensão para arquivos PostScript semelhante ao Pdf,

php5-pspell - módulo para funções Pspell em scripts Php,

php5-recode – pacote que fornece módulo para recodificação de conjunto de caracteres,

php5-snmp - módulo para funções Snmp em scripts Php,

php5-sqlite – pacote que oferece um mecanismo de banco de dados auto suficiente em scripts Php, dispensando uma instalação completa do servidor Sql,

php5-tidy – fornece módulo de extensão libtidy para scripts Php,

php5-xmlrpc - módulo para funções XML-rpc em scripts Php,

php5-xsl – fornece módulo para XSL usando analisador libsltXsl,

php5-json – pacote virtual que fornece Json.

Após instalado os pacotes php deve ser feita a reinicialização do Apache para que ele atualize suas configurações.

3.4. INSTALAÇÃO DO GERENCIADOR DE BANCO DE DADOS *PHPMYADMIN*

Já o *PhpMyAdmin* não necessita de instalação, então é feito o download no endereço:http://www.*PhpMyAdmin*.net/home_page/downloads.php, e descompactado no diretório /var/www. Em seguida, o diretório descompactado foi renomeado para '*PhpMyAdmin*', para simplificar o acesso a ele pelo navegador,

3.5. INSTALAÇÃO DO DRUPAL

Depois de instalado o servidor web, no caso Apache, o banco de dados, MySQL, o interpretador da linguagem, PHP, o gerenciador de banco de dados, *PhpMyAdmin*, o CMS de nossa escolha pôde ser instalado, no caso, o Drupal. O Drupal pode ser adquirido no web site http://Drupal.org/Drupal-7.19-release-notes. Depois de efetuado o download, ele deve ser descompactado na pasta /var/www e renomeado para "Drupal".

Para a instalação é necessário acessar o endereço do diretório já renomeado anteriormente, por meio de qualquer navegador. Sendo esse encontrado pelo caminho de rede que indica o diretório raiz do Apache, ou seja, onde encontram-se os arquivos do Drupal. As instruções para instalação dadas pelo próprio Drupal são intuitivas, ou seja, ajudam o usuário em todo o processo de instalação, onde um usuário leigo pode executar os passos facilmente. Apenas no momento de

configurar o banco de dados, é necessário criar-lo fazendo isto através do *PhpMyAdmin*. O nome da base de dados deve ser informado na instalação do Drupal juntamente com o usuário administrador e a senha.

3.6. INSTALAÇÃO DO *MEMCACHED*

A implementação *Memcached*, que tem duas etapas: a primeira instalar o *Memcached* na distribuição Linux e a segunda instalar o módulo do *Memcached* para Drupal. Primeiramente, o *Memcached* é instalado juntamente com suas dependências via apt-get, *Memcached* - que é o pacote principal, e olib*Memcached*-tools – biblioteca cliente para o servidor *Memcached*. Para instalar a extensão Php para *Memcached* e PECL:

php5-dev – pacote necessário para compilação de módulos adicionais,

php-pear – pacote que contem as classes Pear para Php,

php5-mem*cache* – módulo de extensão *Memcached* para Php5,

após instalados, também, é necessário instalar o PECL no *Memcached*.

O módulo utilizado é uma API (*ApplicationProgramming Interface* – Interface de programação de aplicativos) de integração entre o *Memcached* e o Drupal. Esse módulo, após devidamente configurado e habilitado no Drupal, interfere em toda a comunicação entre o CMS e o banco de dados. Toda primeira requisição para determinada chave feita ao banco, o módulo intercepta essa chamada e faz uma cópia em seu *cache* (na memória RAM do servidor), e toda segunda requisição para a mesma chave, não é realizada com acesso ao banco, mas sim em seu *cache*.

3.7. CONFIGURAÇÃO DO *MEMCACHED*

Com o *Memcached* e seus requisitos já instalados, é necessário também configurá-lo junto ao PHP. Para realizar essa operação, pode ser criado um arquivo chamado mem*cache*.ini no diretório /etc/php5/conf.d. Dentro deste arquivo deve ser colocado o código: “`extension=mem`*`cache`*`.so”,` que permite a utilização da API do *Memcached* na linguagem PHP.

Também foi modificada a seguinte linha no arquivo de configuração. Essa alteração foi realizada visto que o Drupal exige que a estratégia de geração das

chaves de HASH do mem*cache*:"mem*cache*.hash_strategy="consistent". Deste modo, para que sejam feitas de forma consistente podendo assim servidores serem adicionado ou removidos sem interferência nas chaves criadas.

É possível definir o tamanho em megabytes para o *cache*, acessando o arquivo *Memcached*.conf. No diretório “etc”, altera-se o valor da linha abaixo do marcador “#memory”. Nesse momento da configuração novamente é necessário, para atualização das configurações, reiniciar os serviços *Memcached* e Apache.

3.8. CONFIGURAÇÃO DO *MEMCACHED* PARA UTILIZAÇÃO COM DRUPAL

Para ativar o módulo *Memcached* no Drupal, primeiramente deve-se incluir algumas linhas de código no arquivo settings.php localizado no diretório “default” dentro da instalação do Drupal:

```
"$conf['cache_backends'][]='sites/all/modules/contrib/memcache/memcache.inc";
$conf['cache_default_class']='Memcachedrupal';
$conf['memcache_key_prefix']='Chave_Unica';".
```

No lugar de “Chave_Unica”, pode ser colocada qualquer palavra. Essa chave serve para quando o *Memcached* é usado em mais de um site no mesmo servidor. É utilizada uma chave diferente, definida para cada site a fim de definir o registro armazenando diferenciando os registros de cada site.

Feito isso é possível instalar o módulo *Memcached* no Drupal, primeiramente fazendo o download do módulo a partir deste link: http://Drupal.org/node/1410218, e descompactá-lo na pasta modules, encontrada no caminho /var/www/Drupal/sites/all/modules. Por fim, o acesso do Drupal via navegador, na aba módulos, é necessária somente a ativação do módulo *Memcached*, e a partir deste ponto, o *cache* do site já estará sendo gerado automaticamente.

4. RESULTADOS

Foram realizados alguns testes para comprovar a redução do tempo de resposta com a utilização do banco de dados do Drupal em *cache*. Para isto foram utilizadas funções da ferramenta *JMeter*, relacionadas ao teste de carga. Aplicaram-se os testes, com o mesmo site, em duas etapas: a primeira com a função de *Memcached* desativada e a segunda com o *Memcached* ativo, a fim de relatar as diferenças em tempo de resposta de requisições HTTP.

Foi configurado o *JMeter* para realizar quarenta (40) requisições, com um intervalo de um segundo entre elas. O gráfico da figura 7 é resultado desta operação, mostrando uma relação entre o tempo decorrido do teste e o tempo de resposta, denominada latência. No eixo x (horizontal) do gráfico, encontram-se as medições, feitas a cada quinhentos milissegundos. No eixo y (vertical), encontra-se o tempo de resposta das requisições em milissegundos. A latência é obtida pelo cálculo do tempo a partir do final do envio da requisição ao início da resposta do servidor.

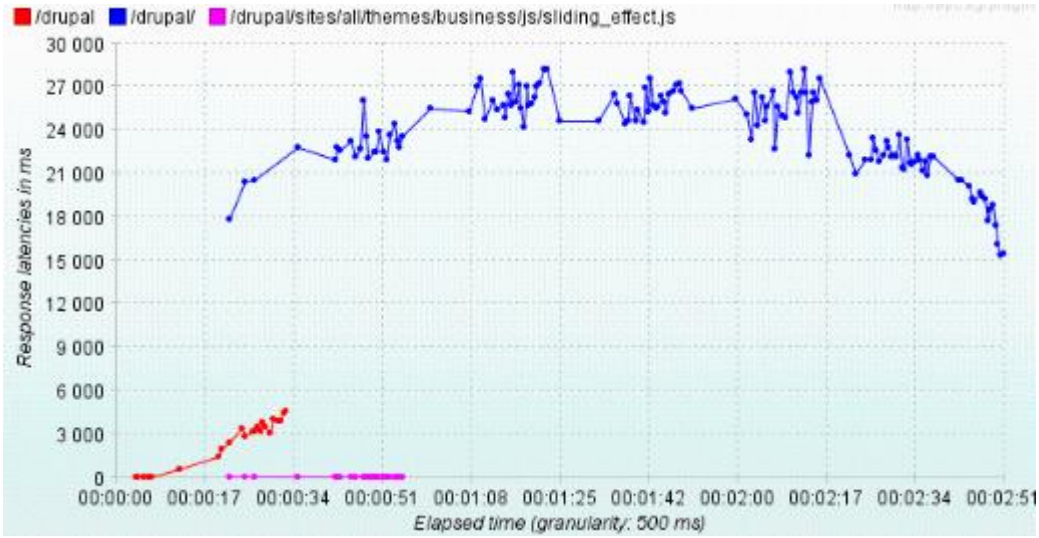

Figura 7 - Gráfico indica tempo de resposta sem utilização de *Memcached*.

Fonte: Autoria Própria.

Na figura 7, a linha em roxo se apresenta linear, pois representa um Javascript[3] do site, o qual não tem a interferência do *cache*, pois este script não tem relações com o banco de dados, então não será relevante para as análises. A linha em vermelho representa um redirecionamento automático do diretório raiz do Drupal para a raiz da página principal da aplicação, o que também não será relevante na análise. A linha em azul representa todas as requisições HTTP para a aplicação em si, onde as requisições do site ao banco são realizadas.

Pode-se observar que sem a utilização do *cache*, as requisições, representadas nas linhas azuis, oscilam entre quinze (15) segundos do menor tempo até cerca de vinte e nove (29) segundos, da resposta mais demorada. Sendo realizado o teste novamente, porém desta vez com o *cache* ativo no sistema, foi obtido o gráfico da figura 8.

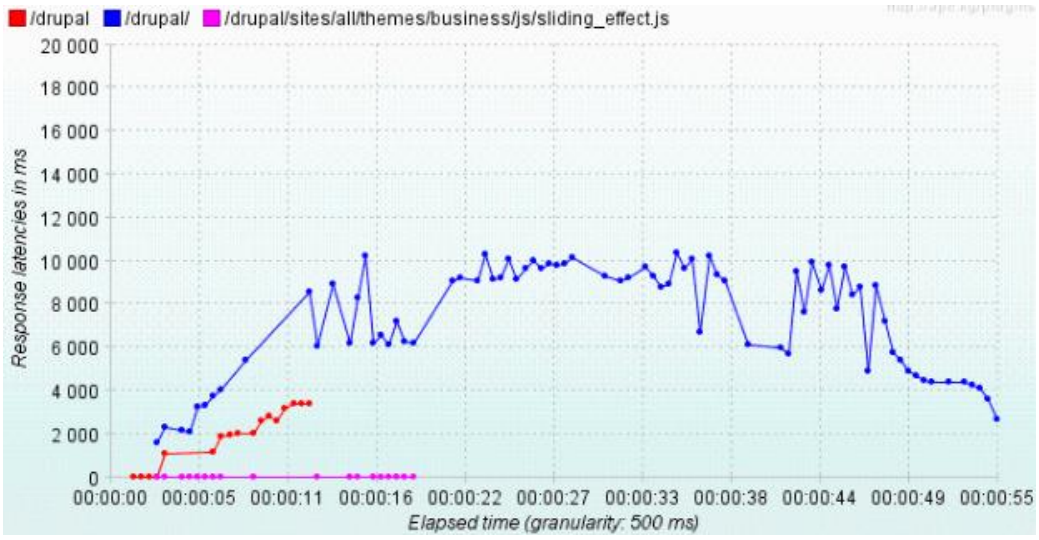

Figura 8 - Gráfico indica tempo de resposta com utilização de *Memcached*.

Fonte: Autoria própria.

Nesta segunda etapa pode-se observar visivelmente que o indicador azul teve um percurso mais baixo em relação ao resultado do teste anterior. Isso significa uma queda considerável no tempo que a aplicação levou para responder as requisições.

Observando a figura 7, que a requisição retornada mais rapidamente levou menos dois segundos e a requisição mais lenta levou pouco mais de dez segundos.

[3] Javascript é uma linguagem com compatibilidade Java e são amplamente utilizados em páginas web

Sendo assim, pode-se observar a melhora significativa na performance da aplicação. O tempo mais lento de resposta do *Memcached* (dez segundos) ainda é mais rápido do que a requisição devolvida em menos tempo quando não é utilizado *Memcached* (em torno de quinze segundos).

Assim foi atestado que utilizando os mesmos parâmetros de teste e mesmas condições, o uso do *Memcached* se mostrou eficaz para seu propósito de diminuição na latência de requisições HTTP.

5. CONCLUSÃO

Como a importância do crescimento das aplicações web no âmbito mundial, cada vez é mais importante o uso das novas tecnologias de aperfeiçoamento, para manter as aplicações competitivas e úteis.

Assim sendo, a utilização de ferramentas como o Drupal e o *Memcached*, auxiliam no desenvolvimento de aplicações web, e na redução do tempo de resposta de requisições HTTP. Com isso o trabalho do usuário e do desenvolvedor torna-se mais ágil, rápido e menos suscetível a falhas.

Este trabalho teve como objetivo principal demonstrar a redução de um dos gargalos entre as aplicações web, de um modo geral com relação ao banco de dados, tornando assim o processo de resposta mais rápido.

Nos estudos realizados foi utilizada a ferramenta *JMeter* para elaborar gráficos e observar a visível melhora na velocidade de retorno de respostas nas requisições HTTP, assim fazendo um comparativo entre o uso das mesmas condições de testes e somente modificando o fator do *cache* sendo utilizado ou não.

Baseado em testes apresentados anteriormente foi possível concluir que o uso de artefatos que diminuem os gargalos de aplicações são essenciais atualmente para aumentar a velocidade com que as aplicações retornam suas respostas aos usuários, pois os mesmos sempre estão em busca de serviços que sejam oferecidos com consistência e rapidez.

5.1 TRABALHOS FUTUROS

Para uma otimização no desempenho de aplicações web, sugere-se futuramente o uso de outras tecnologias de aperfeiçoamento, como por exemplo o Hip Hop PHP e o NGinx.

O Hip Hop PHP é uma ferramenta que faz a transformação de códigos PHP, em arquivos compilados C++. Desse modo, juntando os pontos positivos de cada ferraenta, a linguage PHP teria uma maior flexibilidade no desenvolvimento, enquanto C++ tem maior velocidade de execução.

A ferramenta NGinx é um servidor Proxy HTTP reverso, leve e rápido, auxilia na diminuição de recursos de hardware e pode ser usado juntamente com o servidor Apache para diminuir sua carga de arquivos estáticos.

Portanto, indica-se o uso conjunto dessas ferramentas e bibliotecas com os *scripts* PHP, buscando eliminar os gargalos das aplicações web.

REFERÊNCIAS

THE PHP GROUP. **Manual:** Mem*cache*. Disponível em: <http://php.net/manual/en/book.mem*cache*.php>. Acesso em: 06 dez. 2012.

RAFAEL SILVA. **Manual:** Drupal. Disponível em: <http://Drupal-br.org/manual>. Acesso em: 06 dez. 2012.

MARTIN BROWN. **Aplicando o *Memcached* para aumentar o desempenho do site:**Reduza leituras dos bancos de dados e origens de dados. Disponível em: <http://www.ibm.com/developerworks/br/opensource/library/os-*Memcached*/index.html>. Acesso em: 01 fev. 2013.

TESTEXPERT (Org.). **Ferramenta de Testes: *JMeter*.** Disponível em: <http://www.testexpert.com.br/?q=node/890>. Acesso em: 29 jan. 2013.

ORACLE (Org.). **Identificação rápida de gargalos:** Uma forma mais eficiente de realizar testes de carga. São Paulo: Oracle do Brasil Sistemas Ltda, 2009. 13 p.

CHANDLER, David M.; KRIKNER, Bill; MINATEL, Jim. **Como montar o seu site na Word wide web**. Rio de Janeiro: Campus, 1996. 511 p.

ALVAREZ, Miguel Angel. **Oque é um CMS**. Disponível em: <http://www.criarweb.com/artigos/o-que-e-um-cms.html>. Acessado em: 02 mar. 2013.

SANTOS, Misael da Silva. **Uma Proposta para a Integração de Modelos de Padrões de Software com Ferramentas de Apoio ao Desenvolvimento de Sistemas.** 2004. 105 f. Dissertação (Mestrado) - Curso de Mestrado em Ciência da Computação, Universidade Federal do Ceará, Fortaleza, 2004.

Prieto-Diaz R. **Classification of reusable modules, in Software Reusability**: Concepts and
Models, T.J. Biggerstaff and A.J. Perlis, Editors. Addison-Wesley Pub. Co.: New York,NY; 1989. p.99-123.

DRUPAL PARÁ (Org.). **Módulos do Drupal.** Disponível em: <http://www.Drupal-pa.org/content/m%C3%B3dulos-do-Drupal>. Acesso em: 03 mar. 2013.

PRESSMAN, Roger S.. **Engenharia de Software:** Uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011. 780 p.

FOUNDATION, Apache Software (Org.). **O que posso fazer com ele?** Disponível em: <http://*JMeter*.apache.org/>. Acesso em: 10 mar. 2013.

LAUDON, Kenneth C.; LAUDON, JancePrice. **Sistemas de informação com internet.** Rio de Janeiro: LTC, 1999.

MEYER, Bertrand. **Software object-oriented programing.**New Yourk: Prentice Hall, 1997. 1254 p.

Korth, H.F. e Silberschatz, A.; **Sistemas de Bancos de Dados**, Makron Books, 2a.edição revisada, 1994.