ORPIS: Uma ferramenta livre para sincronização de conteúdo em servidores *Web* distribuídos

Cristiano Cachapuz e Lima[1]
Cláudio Fernando Resin Geyer[2]

Resumo

Uma alternativa de proporcionar mais desempenho à *World-Wide Web* (WWW) é utilizar servidores *Web* distribuídos e replicados (*cluster Web*). Um problema central a ser resolvido em cluster *Web* é como replicar e sincronizar o conteúdo entre os computadores. Esse trabalho pretende abordar exatamente esse problema, e tem por objetivo apresentar uma ferramenta baseada em software livre de replicação e sincronização de conteúdo em servidores *Web* distribuídos.

1 Introdução

Desde sua introdução a *World-Wide Web* (ou simplesmente *Web*) evoluiu de um modelo simples de arquitetura cliente-servidor para um modelo distribuído sofisticado. Essa evolução foi ocasionada devido aos problemas de escala associados com o crescimento exponencial de diversas aplicações anteriormente inexistentes: bibliotecas digitais, educação à distância, áudio e vídeo sob demanda e comércio eletrônico. Essas aplicações ocasionaram um aumento enorme do tráfego na Internet. Alguns sítios *Web* populares recebem milhões de acessos por dia, resultando em tempos de resposta extremamente altos. Quando há um grande aumento na taxa de solicitações aos servidores *Web*, além de sua capacidade, os tempos de resposta e erros na conexão aumentam significativamente. A sobrecarga pode acontecer devido à saturação da largura de banda, do processador ou da memória principal do servidor *Web* ou, até mesmo, da redução da capacidade de conexão do servidor à rede. A lista de pedidos TCP (*Transfer Control Protocol*) do servidor pode ficar sobrecarregada, ocasionando uma degradação no desempenho (ABDELZAHER; BHATTI, 1999).

[1] cristiano@urcamp.tche.br Bolsista URCAMP
[2] geyer@inf.ufrgs.br

Dessa forma, soluções são necessárias para atender os pedidos de maneira eficiente e com desempenho admissível. Uma das soluções possíveis é o uso de servidores *Web* distribuídos. Essa abordagem espalha a carga de pedidos HTTP entre vários computadores conectados atuando como um só com o objetivo de proporcionar um melhor desempenho: um *cluster Web*. Um servidor *Web* distribuído exporta um nome lógico único e o informa para o mundo externo. Cada um dos componentes do *cluster* possui uma réplica do conteúdo a ser oferecido por esse servidor *Web*. Esse *cluster* pode estar instalado fisicamente em um mesmo local ou distribuído geograficamente em diferentes pontos da Internet. Um dos aspectos a serem considerados nessa abordagem é a política de sincronização da atualização das réplicas dos dados no *cluster Web*. Quando da atualização das páginas de um dos servidores, os outros componentes do *cluster* devem refletir exatamente o mesmo conteúdo.

Este trabalho propõe uma ferramenta que permite a manutenção da consistência do conteúdo de servidores *Web* distribuídos e replicados geograficamente – *cluster Web*, com características de transparência e autonomia. Essa ferramenta foi denominada ORPIS (*One Replication Protocol for Internet Services*).

2 ORPIS: concepção e modelagem

A ferramenta ORPIS oferece um módulo a ser anexado ao servidor *Web* que se encarrega de desviar os pedidos para o servidor mais adequado para atender às requisições. O conteúdo (objetos *Web*) é replicado entre os diversos servidores componentes do *cluster Web* distribuído ("comunidade ORPIS"). Um módulo encarrega-se de detectar as atualizações feitas nos objetos *Web*, baseando-se em políticas de intervalos de tempo. Tendo detectado as atualizações feitas, inicia-se o processo de sincronização entre as réplicas. O modelo ORPIS é baseado na forma de replicação descrita em (WIESMANN; PEDONE; SCHIPER, 1999) como multiprimária, pois permite que o cliente envie as requisições para qualquer membro do *cluster*. Essa forma de replicação permite que qualquer uma das réplicas existentes possa ser eleita como a primária. As atualizações no conteúdo do sítio *Web* também podem ser feitas em qualquer servidor distribuído. A não existência de um servidor central permite essa liberdade. O modelo funciona com o conceito de **proprietários** de páginas. O conteúdo total do *cluster Web* é replicado em todos os componentes do *cluster*, porém a atualização de um objeto nas páginas somente é autorizada pelo seu proprietário, tendo ele feito a atualização em qualquer uma das réplicas. Não é permitido que o proprietário de uma página atualize páginas pertencentes a outro proprietário. Esse pressuposto impede a existência de conflitos entre as réplicas. O servidor *Web* que tem instalado o componente Gerente de Replicação do modelo ORPIS, ao detectar uma atualização em seu sistema de arquivos, passa, então, a ser o primário, responsável pela atualização dos demais membros, enviando mensagens de atualizações através do mecanismo de transporte de dados. A troca de informações a respeito de atualizações do conteúdo é realizada entre as réplicas através de trocas de mensagens do

servidor primário com as réplicas *backups*, até que todos os servidores estejam consistentes. Os servidores primários são chamados de nodos propagadores, e os servidores *backups* são denominados nodos receptores. O modelo ORPIS usa uma estratégia de propagação otimista porque não existe sincronismo no envio das atualizações para as réplicas.

A figura 1 apresenta os principais componentes do modelo ORPIS, que são executados no nível do usuário, sem necessidade de reconfiguração do sistema operacional ou do servidor *Web*.

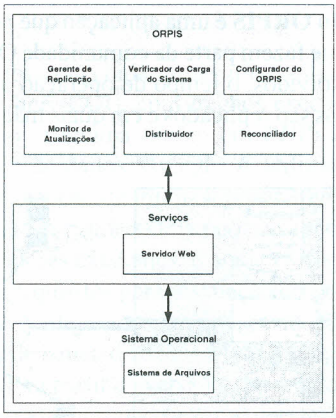

Figura 1: Componentes do modelo ORPIS

O **Verificador de Carga do Sistema** analisa a fila de requisições e calcula a carga de requisições que atualmente está sendo imposta ao servidor em um determinado momento. Uma forma simples de decidir se o servidor está sobrecarregado é através da monitoração do tempo de resposta das requisições ao servidor. Esse tempo de resposta é obtido através do envio de requisições HTTP ao servidor e da medição do tempo de resposta às mesmas. Um tempo de resposta pequeno pode indicar que o servidor não está sobrecarregado. Esse módulo somente é acionado pelo Gerente de Replicação quando do início do processo de sincronização, que o utiliza como forma de detectar a carga de trabalho que está sendo imposta no servidor *Web*.

O **Monitor de atualizações** faz varredura passiva no sistema de arquivos indicado e tem como finalidade o monitoramento de inclusões, exclusões e modificações feitas em diretórios, arquivos ou sistemas de arquivos completos.

O processo de sincronização é iniciado pelo **Gerente de Replicação**, em momentos pré-determinados, ou manualmente, através da requisição do usuário. Ele tem a função de acionar o Verificador de Carga. Se o servidor não estiver sobrecarregado, o Gerente de Replicação inicia o Monitor de Atualizações, que faz a varredura do sistema de arquivos. Se há atualizações a serem propagadas, o Gerente de Replicação tem o papel de incluí-las na Fila de

Objetos a Propagar.

O **Distribuidor** é o módulo responsável pelo contato do nodo propagador com o módulo Reconciliador no nodo remoto. No momento da sincronização, o Distribuidor do nodo propagador estabelece a comunicação com o Reconciliador do nodo remoto. O Distribuidor envia a lista de atualizações para o Reconciliador do nodo receptor.

Reconciliador é o módulo responsável pela comunicação nos nodos receptores. Tem por objetivo enviar respostas de conexão e recepção de listas de atualizações, feitas pelo módulo Distribuidor dos nodos propagadores.

O módulo **Configurador do ORPIS** é uma aplicação que permite incluir, excluir e alterar endereços de servidores *Web* que fazem parte da comunidade ORPIS. Suas funções permitem definir: os servidores *Web* envolvidos, o tempo de operação do reconciliador (operações de sincronização) e quais diretórios são replicados e em quais máquinas do *cluster Web* ocorreria a replicação dos mesmos.

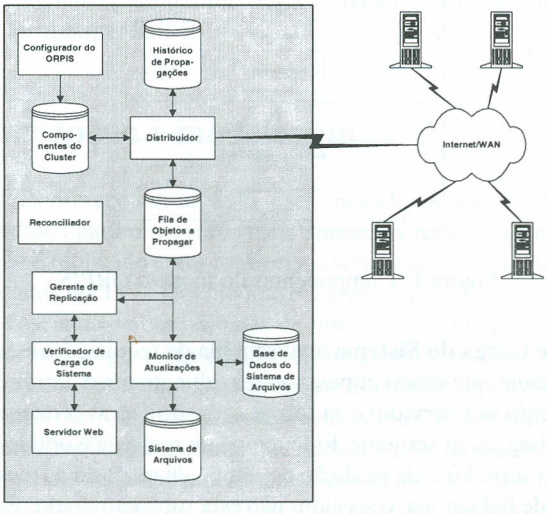

Figura 2: Arquitetura do modelo ORPIS em um nodo propagador

A figura 2 apresenta o diagrama de relacionamentos entre os módulos componentes do modelo ORPIS funcionando em um nodo propagador.

3 Implementação do ORPIS com software livre

O protótipo do modelo ORPIS foi implementado no laboratório de computação da URCAMP, em Bagé. Os equipamentos escolhidos foram dois microcomputadores K6-II, de

500 MHz, ambos com 64 MB de RAM. Como plataforma de sistema operacional, optou-se pelo GNU Linux, com as seguintes características: distribuição Conectiva Linux 6, *kernel* versão 2.2.17-14cl. O servidor *Web* utilizado foi o Apache, versão 1.3.14-6cl. A forma de comunicação adotada entre os módulos do modelo ORPIS é através de *sockets*. O recurso de direcionamento de requisições HTTP foi obtido através da recompilação do servidor *Web* Apache, com a inclusão do módulo `mod_backhand` (SCHLOSSNAGLE, 2000).

3.1 Classes

A classe **Configurador** foi concebida como uma aplicação que se beneficia do modelo *Common Gateway Interface* (CGI), com trechos executados no servidor e no cliente. Foi implementado através um programa escrito em *PHP: Hypertext Preprocessor* (PHP) e HTML, com acesso a banco de dados MySQL. A utilização do configurador do ORPIS deve ser iniciada através da execução de um navegador *Web*. Deve-se informar o endereço *http://nome_do_servidor/orpis/*. A primeira tela permite a inclusão de novos servidores, modificação ou remoção de servidores existentes, e ainda existe uma opção de visualizar todos os servidores cadastrados. Os requisitos para instalação do protótipo do configurador ORPIS são os seguintes componentes configurados em um servidor Linux: servidor *Web* Apache (a partir da versão 1.3), servidor de banco de dados MySQL (a partir da versão 3.23) e módulos do interpretador de scripts PHP (a partir da versão 4.04). A classe **Distribuidor** é responsável por informar aos diversos componentes do *cluster* as propagações que devem ser efetuadas. Ela tem a responsabilidade de conectar um nodo propagador a um nodo receptor e informá-lo sobre quais objetos deverão ser sincronizados. A classe **Gerente** é responsável por gerenciar o processo de propagação do conteúdo para os demais componentes do *cluster*. A classe **Reconciliador** é encarregada de efetuar a comunicação nos nodos receptores. Tem por objetivo enviar respostas de conexão e recepção de listas de atualizações, feitas pela classe Distribuidor do nodo propagador. A comunicação entre essas duas classes é implementada através de mensagens via *sockets*. A classe **Verificador** faz o teste de carga no servidor *Web* do componente do *cluster*. A classe **Monitor** tem a função de fazer a varredura passiva do sistema de arquivos, buscando atualizações que tenham sido feitas. Essa verificação é feita através da comparação dos objetos armazenados no sistema de arquivos com os que estão armazenados na base de dados do sistema de arquivos.

4 Trabalhos relacionados

Durante o desenvolvimento deste trabalho, foi possível identificar e estudar alguns trabalhos e ferramentas cujos objetivos encontram-se dentro do escopo desta pesquisa, isto é, ferramentas de replicação de conteúdo: (a) *Rdist* – seu objetivo é manter cópias idênticas de arquivos em várias estações através dos protocolos remote shell (rsh) ou rcmd; (b) *Rsync*

– pretende ser uma substituta mais rápida e mais flexível ao comando rcp. A ferramenta é baseada no algoritmo rsync, usado para identificar os trechos que são diferentes entre dois arquivos que foram atualizados independentemente; e (c) *Mirrordir* – faz o espelhamento de uma árvore de diretórios em todos os seus detalhes. Também implementa seus próprios *sockets* seguros para transferência de arquivos com encriptação forte.

5 Conclusão

O modelo ORPIS adota uma estratégia de propagação otimista porque não existe sincronismo no envio das atualizações para as réplicas. Outro ponto a ser salientado

é a compatibilidade do modelo ORPIS aos servidores *Web* existentes, sem a necessidade de modificação em suas configurações. Essa característica também se aplica aos editores e atualizadores dos diretórios do servidor *Web*. O uso de software de código aberto no desenvolvimento do protótipo proporcionou um rápido acesso às ferramentas necessárias (sistema operacional, linguagens e gerenciador de banco de dados), com possibilidade de alteração nos códigos fonte como uma alternativa de customização. Os vários componentes do modelo ORPIS ainda estão em desenvolvimento e várias funcionalidades podem ser incorporadas futuramente. Portanto, podem ser destacados como trabalhos futuros: (a) desenvolvimento de um módulo anexado ao servidor *Web* que desvie as requisições dos usuários para o servidor replicado que apresente as melhores condições de acesso ou que esteja mais disponível, baseado em métricas relativas ao desempenho da rede e à carga imposta no momento; (b) aperfeiçoamentos no protótipo, através da otimização do código ou até mesmo geração de código nativo da plataforma de hardware, proporcionando um melhor desempenho e (c) geração de carga de trabalho sintética, simulando o acesso simultâneo de vários clientes e medição do desempenho do servidor *Web*, com o objetivo de simular situações possíveis de acontecerem em um ambiente de produção.

Referências

ABDELZAHER, T.; BHATTI, N. Web content adaptation to improve server overload behavior. *International WWW conference, Toronto, Canada*, dez. 1999.

SCHLOSSNAGLE, T. mod_backhand: A load balancing module for the apache web server. In: APACHECON2000. *Proceedings of the ApacheCon 2000*. Orlando (Florida), 2000.

WIESMANN, M.; PEDONE, F.; SCHIPER, A. A systematic classification of replicated database protocols based on atomic broadcast. In: BROADCAST ESPRIT WG 22455. *Proceedings of the 3rd Europeean Research Seminar on Advances in Distributed Systems (ERSADS'99)*. Madeira Island (Portugal), 1999. p. 264–274.