Web Farm com GNU/Linux num ambiente universitário

Jerônimo A. Bezerra

Centro de Processamento de Dados - Universidade Federal da Bahia (UFBA)
Salvador - BA - Brasil

jab@ufba.br

Abstract. *This article describes the experience gained to install a WEB environment with redundancy, distributed and prepared to receive a lot of simultaneos requisitions, with the security needed, using just free software.*

Resumo*. Esse artigo descreve a experiência adquirida para a instalação de um ambiente WEB redundante, distribuído e voltado para receber grandes quantidades de acessos simultâneos, sem deixar de ser seguro, usando somente softwares livres.*

1. Introdução

Com a popularização dos serviços de *Internet*, principalmente devido ao *WWW* (*World Wide Web*), a disponibilidade dos serviços está cada dia mais importante em todos os ambientes, seja acadêmico ou empresarial. Hoje, o site de cada empresa ou universidade é a imagem da mesma perante o mundo, bem como os serviços que são prestados. Hoje, uma grande variedade de serviços são prestados através da *Internet*, muitos baseados no protocolo *HTTP* (Protocolo de Transferência de Hipertexto), como por exemplo, aplicações de *e-business*, *chats*, ensino à distância, *webmails* e até mesmo algumas transmissões de vídeo. Dada a essa grande variedade de serviços cada vez mais importantes no dia-a-dia das pessoas, os serviços devem estar cada vez mais disponíveis ao usuário final, e o sistema operacional *GNU/Linux* tem se mostrado como uma excelente opção para tal, por sua flexibilidade, estabilidade, segurança e grande aceitação nos mais variados ambientes, principalmente no ambiente acadêmico.

Só para demonstrar o crescimento das aplicações e sites *Web*, segundo uma pesquisa realizada pela NetCraft[1], empresa prestadora de serviços para *Internet*, houve um crescimento de 181.000 sites em novembro de 2005, com relação a mesma pesquisa realizada em outubro de 2005. No total foram 74.572.764 sites que responderam a busca realizada na pesquisa. O crescimento da quantidade de sites é acompanhado pelo aumento das requisições a conteúdos fornecidos pelos serviços Web. Em consequência da grande quantidade de requisições, algumas aplicações demandam um alto desempenho do sistema onde está inserido, e nos casos dos ambientes universitários, podemos notar esse crescimento principalmente nas épocas de inscrições e resultados dos vestibulares.

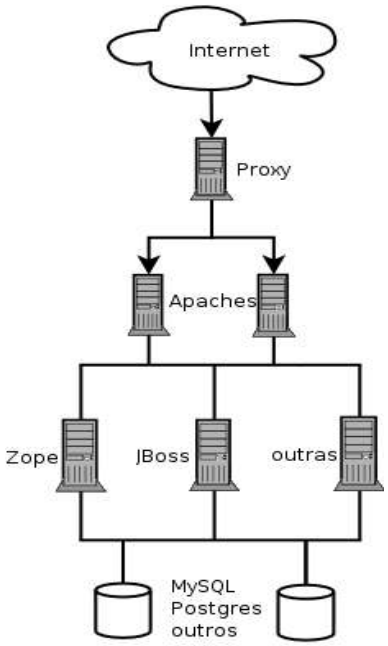
As altas taxas de requisições simultâneas podem ocasionar um aumento do tempo de resposta de atendimento de uma requisição, que ao alcançar um valor muito alto, pode levar a insatisfação do usuário, prejudicar a imagem da organização e, em alguns casos, resultar em perdas financeiras.

Baseado nessas informações, esse artigo descreve uma solução para dificultar quaisquer gargalos nas aplicações, e principalmente, não permitir que caso uma aplicação seja super-utilizada, esta influencie na qualidade das demais envolvidas no ambiente Web da Universidade. A idéia é a construção de uma *Web Farm*, ou seja, uma fazenda de servidores Web. A idéia consiste em manter uma única máquina (com outra em *stand-by* para redundância) fazendo frente aos usuários, e a partir das requisições dos mesmos, encaminhar essa requisição para o servidor responsável. Porém, nesse artigo, esse servidor responsável por fazer o primeiro contato não será apenas um roteador de requisições, e sim um *proxy Web*. A escolha do *proxy* se justifica devido a maior parte do conteúdo dos sites serem estáticos, ou seja, figuras e páginas HTML estáticas. Os servidores de aplicação estarão em hardwares diferentes, permitindo uma otimização do sistema operacional para a função deste, como também isolando as aplicações que consomem muitos recursos. A escolha de usarmos um servidor para Proxy Web com outro servidor em stand-by se justifica

pelo fato de na maioria dos casos, a universidade poder disponibilizar apenas um servidor de alta capacidade de processamento dedicado a esta tarefa, sendo o servidor de stand-by usado apenas enquanto o servidor principal estiver em uma possível manutenção. Porém, caso seja necessário, os dois servidores podem trabalhar paralelamente, fazendo um balanceamento de carga, justificando melhor o investimento realizado, ou até mesmo para aproveitar computadores de menor porte computacional.

2. Arquitetura proposta para o Web Farm

A cenário que será demonstrado se baseia na figura abaixo:



Descrevendo o caminho lógico de uma requisição WEB, seria o seguinte:

1 - O usuário digita no seu navegador a URL desejada, e após a resolução dos nomes por parte dos servidores DNS, ele é redirecionado pela rede ao roteador principal da Universidade e encaminhado para o Firewall;
2 - O firewall redirecionará todo o tráfego Web para o servidor *Proxy Squid*[2], que caso tenha o site solicitado em cache, responde ao usuário, caso contrário encaminha para o servidor Web Apache master;
3 - Ao receber a requisição, o servidor Apache, caso seja conteúdo de sites dinâmicos em PHP ou CGI trata ele mesmo a requisição, caso contrário, ele mesmo faz uma solicitação através dos módulos *rewrite* ou *jk* aos servidores de aplicação, *Zope* ou *JBoss* respectivamente.
4 - Caso alguma das aplicações acessada, ou mesmo algum código PHP faça uso de algum banco de dados, essas aplicações se conectam diretamente nos Servidores Gerenciadores de Banco de Dados (SGBD), que nesse caso podem ser ou *MySQL* ou *PostgreSQL*.

Para monitorar os serviços e verificar quais estão em execução foram utilizados o Mon e o Heartbeat. O Mon é um processo em execução que monitora os serviços configurados e caso algum não esteja em execução, ele ativa o Heartbeat. O Heartbeat também é um processo que fica testando comunicação com o outro servidor monitorado para saber se o mesmo está funcionando. Caso ele verifique que o outro servidor parou de responder e ele consegue se comunicar com um terceiro qualquer, ele assume o IP do outro servidor, e faz um ARP gratuíto, isso é, envia pacotes ARP para os equipamentos atualizarem suas tabelas e encaminhar os pacotes para o servidor certo. Esse processo também é executado se o Mon ativar o Heartbeat, indicando que um serviço está fora e o outro servidor deve então assumir os serviços deste.

Esse modelo de camadas evita em grande parte o acesso do usuário direto às aplicações, que geralmente são mais vulneráveis que os servidores WEB. Algumas filtragens já seriam feitas no servidor Proxy e outras no servidor web. Nesse cenário, os servidores todos seriam redundantes, usando o conceito de alta disponibilidade por *Fail Over*, isto é, quando um servidor fica indisponível, o outro assume seu lugar. Isso permite que tenhamos uma disponibilidade de 5 noves, ou 99,999%, o que é o desejado para a prestação de serviços pela Internet. Outro fator importante nesse modelo é a distribuição das aplicações por máquina, consolidando serviços por servidor. Isso impede que caso uma aplicação seja super-utilizada ou comprometida em sua segurança, que esta comprometa a qualidade dos outros serviços, que em alguns casos, não possuem nenhuma relação entre sí.

3. Softwares utilizados

Os softwares escolhidos para montar esse cenário foram todos softwares livres, alguns já com grande aceitação no mercado e nos ambientes universitários. São eles:

1 - sistema operacional: Debian GNU/Linux[3];
2 - servidor WEB: Apache [4], que segundo a pesquisa do site do Netcraft é responsável por hospedar 70% dos sites da Internet;
3 - servidor Proxy: Squid;
4 - servidores de aplicação: Zope[5] e JBoss[6];
5 - servidores de banco de dados: MySQL[7] e PostgreSQL[8];
6 - softwares para fazer a redundância: Mon[9] e Heartbeat[10];

Fora essas ferramentas ligadas diretamente ao propósito macro do sistema, serão usadas as seguintes ferramentas:

7 - IDS de Host: AIDE para monitorar anormalidades no comportamento do servidor e alterações em arquivos indevidos
8 - NET-SNMP para quantificar e caracterizar o tráfego passante nas placas de rede, bem como monitorar o uso dos recursos do sistema.

4. Hardwares utilizados

Dada a flexibilidade desse ambiente, ele fica muito escalável. Sendo assim, caso haja a necessidade de tornar os servidores como balanceadores de carga em vez de apenas *Fail Over*, é possível fazê-lo sem muita dificuldade. Todas as aplicações envolvidadas possuem algum tipo de redundância, inclusive os servidores de aplicação *Zope* e *JBoss*. Sendo assim, o hardware necessário é proporcional à expectativa de uso dos serviços, seja em quantidade seja em qualidade dos hardwares. No ambiente em questão, o servidor Proxy é um computador Dell e os servidores Apache e de aplicação são servidores IBM de pelo menos dois processadores. Todas as máquinas possuem dados armazenados em Storages externos, através de placas Fibre Channel ou SCSI, e possuem placas de rede Gigabit Ethernet.

5. Conclusão

Num ambiente universitário que provê serviços de Ensino à distância, hospedagem de páginas de departamentos e professores, aplicações de gerenciamento de conteúdo, todos geralmente muito acessados, esse ambiente centralizado de serviços Web torna o gerenciamento muito mais fácil por parte dos administradores, apesar de ser aparentemente complexo, dado que além de bem organizado, é redundante em todas as suas camadas. Com a facilidade de gerenciamento, a segurança fica mais eficaz, o que protege ainda mais a imagem da organização e passa mais confiança a seus usuários, para que os mesmos possam delegar suas páginas e sites de projetos aos CPD das universidades e se concentrem no seu foco que são as pesquisas propriemente ditas.

6. Referências

[1] NETCRAFT. Web Server Survey Archives – Disponível em http://news.netcraft.com/archives/web_server_survey.html. Último acesso: 27/02/2006

[2] Squid Cache – Disponível em http://www.squid-cache.org . Último acesso: 27/02/2006

[3] Debian GNU/Linux - Disponível em http://www.debian.org . Último acesso: 27/02/2006

[4] Apache HTTP Server - Disponível em http://httpd.apache.org Último acesso: 27/02/2006

[5] Zope - Disponível em http://www.zope.org . Último acesso: 27/02/2006

[6] JBoss - Disponível em http://www.jboss.org . Último acesso: 27/02/2006

[7] MySQL - Disponível em http://www.mysql.org . Último acesso: 27/02/2006

[8] PostgreSQL - Disponível em http://www.postgres.org . Último acesso: 27/02/2006

[9] Mon - Disponível em http://www.kernel.org/software/mon . Último acesso: 27/02/2006

[10] Heartbeat – Disponível em http://www.linux-ha.org . Último acesso: 27/02/2006