CLUSTER WEB

André Soares Rodrigues Francislane Pereira Milene Moreira de Souza Thiago Maia Gouvêa Alves
andresrodrigues@ig.com.br francis.aedb@gmail.com mimims@terra.com.br thiago.alves@mpsa.com
Associação Educacional Dom Bosco (AEDB), Faculdade de Sistemas de Informação - Resende, RJ, Brasil
Associação Educacional Dom Bosco (AEDB), CPGE -Resende, RJ, Brasil

RESUMO

O surgimento de novas aplicações que utilizam HTTP nas suas transações e a crescente popularidade da World-Wide Web (WWW) provocaram buscas por servidores Web de elevado desempenho. Uma alternativa é cluster Web, isso é, um conjunto de servidores Web distribuídos que espalham a carga de pedidos entre vários computadores atuando como um só, com o objetivo de proporcionar alto desempenho. Esse artigo pretende abordar as características dessa arquitetura e tem por objetivo propor um modelo de replicação de conteúdo em servidores Web distribuídos com características de transparência, autonomia e buscando um alto desempenho.

Palavras-Chave: Cluster Web. Servidores Web. OpenMosix. MigShm.

1. INTRODUÇÃO

Desde sua introdução, no início dos anos noventa, a *World Wide Web* (WWW) evoluiu muito no que diz respeito à sua utilização. Essa evolução foi conseqüência do aparecimento 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 e na WWW. Alguns sites *Web* populares recebem milhões de acessos por dia, resultando em tempos de resposta extremamente altos. Isso frustra muitos usuários e causa a preocupação em muitos administradores *Web*. Um dos principais problemas enfrentados por estes é adequar os recursos para atender às exigências dos usuários. Esse desafio exige que os administradores *Web* estejam aptos a monitorar "gargalos", prever a capacidade de seus sites *Web* e determinar a melhor maneira de resolver os problemas de desempenho causados pelo aumento da carga de trabalho imposta aos seus sites.

O ambiente *Web* possui características particulares em relação a outros sistemas tradicionais distribuídos. Algumas dessas características provocam um impacto considerável no planejamento dos servidores *Web*. Primeiro, o número de clientes, além de estar em constante crescimento, pode chegar à ordem de dezenas de milhões. O comportamento dos usuários durante a navegação é completamente aleatório, impedindo uma correta previsão da carga de trabalho que pode estar sendo imposta em um servidor *Web*. Em certos momentos, os sites estão quase inativos, sem visitantes. De repente, sem qualquer aviso, o tráfego pode aumentar enormemente. 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 da CPU ou da memória principal do servidor *Web* ou, até mesmo, da redução da capacidade de conexão do servidor à rede. Também a lista de pedidos TCP (*Transfer Control Protocol*) do servidor pode ficar sobrecarregada, contribuindo para a degradação do desempenho geral. Dessa forma, soluções são necessárias para atender aos pedidos de maneira eficiente e com desempenho admissível.

Visando a diminuição do alto custo das máquinas paralelas convencionais vem sendo desenvolvida a idéia de Máquinas Paralelas Virtuais (MPV) baseadas em estações de trabalho. Estas utilizam redes de computadores convencionais, locais e/ou remotas, como barramento para o trafego de dados entre computadores seqüenciais para assim viabilizar a paralelização de tarefas. Para tal se faz necessário o uso de um software que permita que um conjunto heterogêneo ou homogêneo de computadores, de qualquer natureza, seja visto como uma única máquina formando assim o que se denomina um Cluster de computadores.

O paradigma OpenMosix oferece uma solução transparente ao usuário sem necessidade de alteração em código-fonte de qualquer aplicativo.

2. CLUSTER COMPUTING

Cluster é o nome dado a um sistema montado com um conjunto computadores, cujo objetivo é fazer com que todo o processamento seja distribuído a todos os computadores da rede, mas de maneira que de a impressão de que existe apenas um computador trabalhado. Com isso, é possível realizar processamentos que até então somente computadores de alto desempenho seriam capazes de fazer.

Cada computador do cluster é denominado nó. Todos devem ser interconectados, de maneira a formarem uma rede, de qualquer topologia, como mostra a Figura 2.1. Essa rede precisa ser criada de uma forma que permita o acréscimo ou a retirada de um nó, mas sem interromper o funcionamento do Cluster. Independente do sistema operacional usado é preciso usar um software que permita a montagem do Cluster. Esse software vai ser responsável, entre outras coisas, pela distribuição do processamento. Esse é um ponto crucial na montagem de um Cluster. É preciso que o software trabalhe de forma que erros e defeitos sejam detectados, oferecendo meios de providenciar reparos, mas sem interromper as atividades do Cluster.

Fig. 2.1 – Cluster Computing.

Para que exista um Cluster, são necessários pelo menos dois computadores. Evidentemente, quanto mais computadores existirem no Cluster, maiores serão os custos de implementação e manutenção. Isso não se deve apenas ao preço dos computadores, mas também pelos equipamentos (switches, cabos, hubs, nobreaks, etc.). Mas ainda assim, os

custos serão menores do que a aquisição e manutenção de computadores poderosos e algumas vezes o processamento é até mais eficiente.

Clusters são usados quando os conteúdos são críticos ou quando os serviços têm que estar disponíveis e processados o quanto mais rápido possível, como os serviços WEB, por exemplo. Utilizar este tipo de máquina tem sido uma saída para substituir os supercomputadores, pois podem oferecer desempenho semelhante e com um custo mais baixo. Algumas vantagens dos clusters são:

- Escalabilidade: é possível aumentar o desempenho do mesmo adicionando ou trocando os microcomputadores que compõem o cluster;
- Tolerância à falhas: o cluster mantém o funcionamento mesmo com a paralisação de alguns nós;
- Baixo custo: utilizam recursos de fácil acesso e de uso comum;
- Independência de fornecedores: principalmente por utilizar microcomputadores comuns (inclusive de plataformas heterogêneas), não estão presos a uma tecnologia específica.

3. SERVIDORES WEB DISTRIBUÍDOS

Uma das soluções apresentadas 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 alto desempenho: um cluster *Web*. Um servidor *Web* distribuído exporta um nome lógico único e o informa para o mundo externo. Do ponto de vista do cliente, que envia um pedido HTTP, o grupo de servidores é apenas um servidor que irá manipular o pedido HTTP da mesma maneira que qualquer outro servidor *Web* faz. A partir do recebimento do pedido, o *cluster* o dirige a um dos membros do grupo para o processamento. 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.

Servidores *Web* distribuídos necessitam mecanismos de decisão do servidor replicado que deverá responder aos pedidos de determinado cliente. A situação ideal é descobrir o servidor replicado mais apropriado para o cliente em questão se comunicar com o melhor desempenho possível. Essa otimização é uma política baseada em decisão, normalmente estabelecida por proximidade, mas também pode ser baseada em outros critérios, como, por exemplo, a carga de pedidos. As formas mais comuns existentes são:

- *Links* de navegação: a maneira mais simples de comunicação entre clientes e réplicas. Esse mecanismo usa URLs individuais dentro das páginas que apontam para os servidores replicados. O cliente seleciona manualmente o *link* do servidor replicado que deseja usar;
- Redirecionamento de HTTP: os clientes são redirecionados para um servidor replicado ótimo através do uso dos códigos de resposta do protocolo HTTP. Por exemplo, 302 "*Found*" ou 307 "*Temporary Redirect*". Um cliente estabelece comunicação com um dos servidores replicados. O servidor replicado contatado inicialmente pode escolher aceitar o serviço ou redirecionar o cliente novamente;
- Redirecionamento através de DNS: o *Domain Name Service* (DNS) oferece um sofisticado mecanismo de comunicação entre clientes e réplicas. Isso é possível pelos servidores DNS que ordenam os endereços IP resolvidos, baseado em políticas de

serviço. Quando um cliente converte o nome de um servidor, o servidor DNS ordena os endereços IP dos servidores replicados iniciando pela melhor réplica e terminando na réplica menos apropriada.

Replicação é a criação e a manutenção de uma cópia duplicada de uma base de dados ou sistema de arquivos em computadores diferentes, os quais são normalmente servidores. Um dos aspectos a serem considerados em servidores *Web* distribuídos é a política de gerenciamento na 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.

A replicação de objetos em sistemas distribuídos, normalmente, é utilizada para torná-los mais confiáveis e seguros, pois o sistema pode sobreviver à falhas de uma ou mais cópias do componente replicado. Nos ambientes de *cluster Web*, o objetivo principal da replicação é permitir o balanceamento de carga entre os componentes, além de permitir tolerância à falhas quando um dos servidores deixa de funcionar. Além disso, é desejável que os detalhes da replicação sejam escondidos dos usuários finais, proporcionando transparência de localização, isto é, o fato de o usuário usar uma réplica sem perceber.

Podem-se distinguir três formas de replicação de conteúdo em servidores *Web* distribuídos:

- Replicação em lotes: o servidor replicado a ser atualizado é quem inicia a comunicação com um servidor original. A comunicação é estabelecida em intervalos baseados em transações enfileiradas que são agendadas para processamento posterior. As políticas de agendamento variam, mas, normalmente, ocorrem em um determinado intervalo de tempo. Uma vez que a comunicação é estabelecida, conjuntos de dados são copiados para o servidor replicado. Os protocolos mais utilizados são FTP e RDIST. São muito usados para sincronização de espelhos de sítios *Web* na Internet;

- Replicação por demanda: os servidores replicados obtêm o conteúdo quando necessário devido à demanda do cliente. Quando um cliente solicita um recurso que não esteja no conjunto de dados do servidor replicado, é feita uma tentativa de atender ao pedido, buscando o recurso do servidor original e o retornando para o cliente que o solicitou. Os protocolos mais utilizados são FTP, HTTP e ICP;

- Replicação sincronizada: os servidores replicados cooperam usando estratégias sincronizadas e protocolos especializados de réplica para manter os conjuntos de dados replicados coerentes. Estratégias de sincronização variam desde fortemente coerentes (alguns poucos minutos) até fracamente coerentes (algumas horas). As atualizações ocorrem entre as réplicas baseadas nas restrições de tempo do modelo de coerência empregado e são feitas, geralmente, apenas através dos dados que foram modificados. Os dois protocolos abertos de replicação sincronizada mais difundidos são AFS e CODA. Além desses, há um outro protocolo proprietário, o Novell NRS. Todos os protocolos conhecidos usam métodos de troca com chave criptográfica forte, que são baseados no modelo secreto compartilhado Kerberos ou no modelo de chaves públicas/privadas RSA. Esse tipo de replicação ainda não é muito difundido, encontrado em apenas alguns sítios, principalmente em centros universitários de pesquisa.

Com conteúdo de bancos de dados relacionais, se não houver replicação, o “gargalo” ocorre no acesso aos dados, principalmente no acesso de escrita, já que as transações de

bancos de dados devem satisfazer os critérios chamados ACID (Atomicidade, Consistência, Isolamento e Durabilidade).

Para garantir a consistência e a durabilidade da transação, deve-se proibir o acesso aos dados que serão replicados no início da transação (isolamento), para que nenhum outro processo possa modificar ou corromper os dados durante essa transação. A transação pode ocorrer até o fim ou ser completamente cancelada (atomicidade). Como os dados de um servidor *Web* formam um conjunto único e com acesso estritamente serial, o tempo de duração da transação fica limitado à rapidez com que se proíbe o acesso, modifica-se a base e se disponibiliza os dados novamente.

Um problema a ser resolvido é como replicar o conteúdo de forma transparente, autônoma e com alto desempenho. Deve-se avaliar, cuidadosamente, que aspectos devem ser privilegiados, como a consistência, a disponibilidade dos dados ou o desempenho que se deseja alcançar. As questões introduzidas em [8] devem ser tratadas ao se trabalhar com replicação no ambiente *Web*: algoritmos de distribuição das réplicas, colocação das réplicas, consistência e atualização de versões.

4. PROPOSTA OPENMOSIX

OpenMosix é uma extensão do kernel do sistema operacional Linux para criação de clusters vistos de maneira transparente pelas aplicações e usuário, com a paralelização de tarefas dando - se a nível de kernel do sistema operacional. Esta extensão faz com que uma rede de computadores seqüenciais possa se tornar um cluster para aplicações Linux. Ao implementar esta extensão, os nós do cluster, começam a se comunicar de forma a adaptar - se ao volume de trabalho.

Uma máquina pode se juntar ou sair do cluster a qualquer momento. Como principal característica o OpenMosix tenta continuamente otimizar o alocamento de recursos. Dado que todas as extensões do OpenMosix estão dentro do Kernel, cada aplicação do Linux automaticamente beneficia- se, de forma transparente, do conceito de computação paralela. A Figura 4.1 mostra como o cluster OpenMosix comporta-se como um sistema multiprocessado distribuindo processos entre os nós do cluster. O OpenMosix consiste de duas partes, o Preemptive Process Migration (PPM) e um conjunto de algoritmos de compartilhamento de recursos implementado na forma de modulo no nível de kernel. O OpenMosix ainda possui um sistema de arquivos paralelo e ferramentas para administração do cluster.

O OpenMosix se beneficia principalmente quando múltiplos processos estão em andamento, ou apenas um trabalho que funcione por muitas horas seguidas, os exemplos para estes tipos de aplicação são numerosos, destes incluem as instituições financeiras que trabalham com a análise do risco ou as instituições de pesquisa científicas que funcionam com base de dados de comparação de DNA.

Há ainda algumas características que faltam ao Openmosix, nem todas as aplicações podem migrar para outros nós, estas tem características que impedem que um processo esteja separado do seu contexto no sistema ou possuam memória compartilhada. Em alguns processos, onde há características firmemente ligadas ao Hardware ou a rede, a migração poderia resultar em diminuição na computação do processo como um todo. Atualmente uma das maiores limitações do OpenMosix é quanto as aplicações que utilizam de memória compartilhada ou sejam multi - threaded, pois este não consegue migra - los de maneira eficiente.

Uma solução para o problema de compartilhamento de memória é o migShm (Migration of shared memory) que tem como objetivo possibilitar à migração de processos,

tais como Web Servers, Sistemas Gerenciadores de Banco de Dados e outros que façam uso de memória compartilhada sem nenhuma perda de perfomance em clusters OpenMosix. Processos que usem as mesmas regiões de memória compar tilhadas podem migrar para outros nós de forma transparente e eficiente.

Dada a necessidade de se manter a consistência entre as diferentes copias das páginas de memória, optou - se por um modelo conhecido como Eager Release consistency model, que implementa um sistema onde uma cópia local da página de memória modificada irá apenas ser escrita no local original quando o lock do processo deste segmento de memória for liberado, assegurando que o nó proprietário da memória compartilhada tenha sempre a cópia mais atualizada. A atualização das páginas de memória no nó local é feito atrás de uma operação de writeback, assim que é liberado o lock do processo no nó remoto.

O migShm é implementado na forma de patch para o kernel do Linux com o OpenMosix e ainda não resolve todos os problemas relacionados com a Memória Compartilhada Distribuída.

Figura 4.1 – Cluster OpenMosix.

5. EXPERIMENTOS COM O OPENMOSIX E MIGSHM

Os experimentos com o OpenMosix e migShm tem como objetivo possibilitar à migração de processos, tais como Web Servers, Sistemas Gerenciadores de Banco de Dados e outros que façam uso de memória compartilhada sem nenhuma perda de perfomance em clusters OpenMosix.

Para tais experimentos podem ser utilizadas máquinas comuns, consideradas até obsoletas. Designa-se um dos nós do cluster que realizará o trabalho de solicitação das requisições ao Apache Web Server para uma máquina do cluster com duas interfaces de rede afim de que o trafégo da requisição ao Apache Web Server não interferir-se na comunicação entre máquinas do cluster. A quantidade de requisições, definida aqui como medida de carga, ao Servidor Web foi variada afim de avaliar a melhoria na escalabilidade na parelização dos processos. Não é necessário ter o Apache Web Server instalado em todas as máquinas do

cluster, apenas naquela que irá receber as requisições. A migração dos processos se realizará de forma transparente não precisando fazer qualquer alteração. Para iniciar o cluster OpenMosix, basta recompilar o kernel e iniciar o daemon para o cluster ser formado automaticamente.

A carga de processos é distribuida (load balance) conforme avaliação do algoritmo presente no OpenMosix que mede tanto a disponibilidade de recursos da rede quanto das máquinas pertecentes a este, tirando uma medida de custo e avaliando se deve e como distribuir a carga de processos entre membros do cluster. A página utilizada para medir a perfomance das requisições ao Apache Web Server é formada de uma página PHP que gera uma série de operações matemáticas e de leitura de disco, demandando tempo para a execução desta, formando assim um cenário propicio para a avaliação de migração de processos.

Uma medida de carga utilizada nos testes é determinado pela quantidade de conexões abertas e de processos por conexão, ao se efetuar uma carga de quatro unidades abre- se quatro conexões e requisita-se quatro páginas por conexão gerando a demanda necessária para avaliação de desempenho.

6. CONCLUSÕES

Existe a necessidade de se oferecer desempenho na WWW, principalmente na área de comércio eletrônico, onde um tempo de resposta elevado pode cancelar uma venda, ocasionando prejuízos às empresas. Uma das formas de atender muitos acessos concorrentes aos sítios *Web* é distribuir a carga de pedidos entre vários servidores, permitindo balanceamento de carga e tolerâncias à falhas, caso um desses servidores venha a falhar. Essa abordagem exige que o conteúdo desse *cluster Web* seja idêntico. Esse sincronismo de conteúdo é obtido através da replicação do conteúdo entre os vários servidores. A literatura de sistemas distribuídos apresenta diversas formas de replicação de arquivos em ambientes distribuídos. Um problema a ser resolvido é como replicar o conteúdo de forma transparente, autônoma e com alto desempenho. Esse trabalho propõe um modelo de replicação de conteúdo em servidores *Web* distribuídos com características de transparência, autonomia e buscando um alto desempenho.

O OpenMosix traz uma nova solução para a construção dos Computadores de Alta Performance e de Baixo Custo com uma abordagem mais transparente ao usuário e altamente eficiente. Muitos aplicativos conseguem se beneficiar sem qualquer alteração de forma e ter um ganho significativo de desempenho.

O migShm juntamente com o OpenMosix permite a migração de algumas aplicações com memória compartilhada, mas ainda encontra – se numa fase imatura de desenvolvimento fazendo com que o sistema se torne instável em dados momentos fazendo com que em muitos aplicativos não funcionem corretamente ou apresentem problemas inesperados. Observamos um sensível ganho de desempenho ao se aumentar à carga de consultas ao Servidor Web em clusters de quatro máquinas.

O barramento de comunicação tem forte influência na perfomance geral de um Cluster de Computadores baseado em Estações de trabalho, mesmo com esta limitação, dada a estrutura dos testes, provou - se uma solução eficiente.

6. REFERÊNCIAS

MAASK, Group. **MigShm Project Report**. Disponível em: <http://mcaserta.com/maask /Migshm_Report.pdf>. Acesso em 30. Julho 2006.

BUYTAERT, Kris. **OpenMosix Howto**. Disponível em : <http://howto.ipng.be/openMosix-HOWTO/>. Acesso em 05. Agosto 2006.

CORREIA, Martinho. **Web services e arquitetura**. Disponível em : <http://dme.uma.pt/ jcardoso/Research/Papers/CAPSI2006-id66.pdf>. Acesso em 05. Agosto 2006.

GOLDCHLEGER, Andrei. **Projeto Integrade.** Disponível em : <http://gsd.ime.usp.br/ publications /thesis/MestradoAndrei.pdf>. Acesso em 05. Agosto 2006.

INFO Wester. **Cluster: Principais Conceitos.** Disponível em: <http://www.infowester.com/cluster.php>. Acesso em: 10 Agosto. 2006.

PITANGA Marcos. **Computação em Cluster**. Disponível em: <http://www.clubedohardware.com.br/printpage/153>. Acesso em: 10 Agosto 2006.