Um Sistema de Arquivos Compartilhado em Nível de Usuário Baseado em Tabelas Hash Distribuídas

Pedro Eugênio Rocha, Luiz Carlos Erpen de Bona

Departamento de Informática – Universidade Federal do Paraná
Caixa Postal 19.081 – CEP 81.531-980 – Curitiba – PR – Brasil

{pedro,bona}@c3sl.ufpr.br

Abstract. *This paper describes DIFS (Distributed Filesystem), a fully distributed filesystem based on a DHT (Distributed Hash Table). Each node provides some amount of storage space to the network, creating a huge shared filesystem available to all nodes. To create a real filesystem in user mode, the system uses FUSE (Filesystem in Userspace). A DHT is used to associate file names to the nodes that store them, as all files and directories created on the network are replicated and stored in a set of nodes. Finally, we present the results of the tests using the prototype developed.*

Resumo. *Este artigo apresenta o DIFS (Distributed Filesystem), um sistema de arquivos totalmente distribuído baseado em DHT (Distributed Hash Table). Cada nodo do sistema cede uma certa quantidade de espaço de armazenamento, formando um grande sistema de arquivos compartilhado. Para criar um sistema de arquivos real e em nível de usuário o sistema utiliza o FUSE (Filesystem in Userspace). Uma DHT é utilizada para associar nomes de arquivos aos nodos que os armazenam, pois todo arquivo ou diretório criado na rede é replicado e armazenado em um conjunto de nodos. Finalmente, apresentamos os resultados dos testes realizados utilizando o protótipo desenvolvido.*

1. Introdução

As redes P2P (*Peer-To-Peer*) vêm sendo muito utilizadas atualmente para o compartilhamento de recursos e informações em larga escala na Internet e em redes privadas. Por serem redes, em geral, altamente escaláveis, auto-organizáveis e que não necessitam de controle centralizado, diversas aplicações de compartilhamento presentes hoje na Internet as utilizam [eDonkey network , Clarke et al. 2001, Kubiatowicz et al. 2000, Muthitacharoen et al. 2002]. Dois tipos de sistemas baseados em redes P2P são importantes a este trabalho: redes de compartilhamento de arquivos e sistemas de arquivos distribuídos. Nas redes de compartilhamento, os arquivos inseridos dificilmente são modificados, e nem todos os nodos precisam possuir permissão de escrita. Os sistemas de arquivos distribuídos precisam lidar com arquivos sendo constantemente modificados, inseridos e excluídos por todos os nodos da rede.

Redes de compartilhamento de arquivos como o eDonkey [eDonkey network] e o Napster Network [Carlsson and Gustavsson 2001] são baseadas em um conjunto de servidores centrais que realizam a indexação dos arquivos. O *download* dos arquivos é realizado diretamente entre os nodos. A estratégia de compartilhar a carga de *download* entre os participantes funcionou muito bem na Internet, permitindo que essas redes alcançassem

dimensões muito maiores que as redes baseadas no modelo cliente-servidor. Entretando, tais redes ainda dependem de servidores.

O NFS (*Network Filesystem*) [Sandberg 1986] e AFS (*Andrew Filesystem*) [OpenAFS], apesar de não serem baseados em redes P2P, são exemplos de sistemas de arquivos distribuídos que possibilitam o acesso remoto a arquivos e diretórios. Ambos são baseados em servidores, apenas possibilitando aos clientes montar sistemas de arquivos exportados por um ou mais servidores remotos. O Ivy [Muthitacharoen et al. 2002] é um exemplo de sistema de arquivos completamente distribuído (não necessita de servidores) baseado em uma rede P2P. Todas as modificações em blocos de dados realizadas no Ivy são armazenadas em *logs*, em uma DHT (*Distributed Hash Table*). Para acessar um arquivo, um nodo deve consultar todos os logs referentes a ele, obtendo assim a sua versão mais atual.

Este trabalho propõe o DIFS, um sistema de arquivos *open-source* totalmente distribuído que provê um espaço de armazenamento compartilhado entre os participantes de uma rede P2P. Todos os nodos cedem uma certa quantidade de espaço de armazenamento para a rede, criando um sistema de arquivos compartilhado disponível a todos os participantes. O sistema armazena réplicas de todos os arquivos inseridos, garantindo a disponibilidade mesmo perante falhas em nodos. Uma DHT é utilizada para realizar a indexação dos arquivos aos nodos que os armazenam. Apresentamos também neste artigo os resultados dos experimentos realizados com o protótipo do sistema.

O sistema proposto utiliza o FUSE (*Filesystem in Userspace*) [FUSE] para prover a interface de um sistema de arquivos convencional em modo usuário. O FUSE é um projeto *open-source* disponível para a maioria dos sistemas *Unix-like*, que permite a implementação de um sistema de arquivos completo através de um módulo carregável do kernel e uma API a ser utilizada pelo programa em modo usuário.

1.1. Tabelas Hash Distribuídas (DHT)

Um dos maiores problemas em redes P2P é a localização de dados. Além de eficiência nas buscas, é preciso também garantir que todos os dados inseridos possam ser encontrados. Existem diversas propostas de redes que cumprem esses requisitos, diferindo essencialmente na forma que os nodos são estruturados ou na maneira como realizam o roteamento. Todos os dados inseridos são armazenados em algum local da rede, podendo ser recuperados a qualquer momento. Essas redes provêem a funcionalidade de uma grande tabela hash, distribuída entre os seus participantes.

As DHT's, como essas redes são chamadas, realizam basicamente a operação: dada uma chave qualquer, retornar o objeto associado (valor). Objetos podem ser arquivos, serviços, dados, ou qualquer outro valor indexado à chave. As DHT's são descentralizadas, e em geral possuem grande escalabilidade, adaptando-se à entrada e saída constante de nodos. Alguns dos principais protocolos de DHT são CAN [Ratnasamy et al. 2001], Chord [Stoica et al. 2001] e Pastry [Rowstron and Druschel 2001].

Nas DHT's, todo nodo ou dado possui uma chave ou identificador único, independente da implementação. O conjunto de todas as chaves possíveis na rede forma um *espaço de chaves*, cujo armazenamento é distribuído entre os nodos segundo algum critério de distribuição. Todo nodo mantém uma lista com um subconjunto de nodos conhecidos nesta rede e seus identificadores. As DHT's possuem também uma noção

abstrata de distância entre chaves, que, em geral, não possui relação com fatores físicos como localização geográfica ou latência da rede entre os nodos.

O restante deste trabalho está organizado da seguinte forma. A seção 2 apresenta em detalhes a proposta do DIFS, na seção 3 são analisados os resultados obtidos nos experimentos com o protótipo do sistema e a seção 4 conclui este trabalho.

2. DIFS

O DIFS é um sistema de arquivos baseado em uma rede P2P totalmente distribuída e estruturada que utiliza uma DHT como rede *overlay*. O sistema é composto por diversos nodos que cedem espaço de armazenamento à rede, cuja capacidade de armazenamento é igual à soma dos espaços cedidos. Cada arquivo criado na rede é armazenado por um conjunto de nodos. Todo arquivo possui um nodo, denominado *keeper*, responsável por gerenciar as réplicas e manter a consistência entre elas. Os nodos que armazenam as réplicas de um arquivo são denominados *storers*. Arquivos de um mesmo diretório ou de um mesmo usuário não são necessariamente armazenados em um mesmo nodo, não existindo restrição alguma quanto à localização física dos arquivos na rede.

Todas as operações que modificam um arquivo ou suas meta-informações devem ser enviadas ao *keeper* do arquivo, para que ele repasse aos *storers* e mantenha a coerência entre as réplicas. Já as operações de leitura, que com grande probabilidade ocorrerão em maior número, podem ser requisitadas a qualquer *storer*. Os diretórios são armazenados da mesma maneira que arquivos normais. No DIFS, por questões de desempenho, os diretórios guardam também os atributos dos arquivos que armazenam. Daqui em diante "arquivo" pode significar um arquivo normal ou um diretório.

O sistema utiliza a DHT para indexar os nomes dos arquivos aos nodos que os armazenam. Para cada arquivo da rede, a DHT armazena o seguinte par (chave, valor): [`nome_do_arquivo, A#B#C#D`]. *A, B, C* e *D* são os endereços dos nodos que armazenam o arquivo referente à chave, sendo o primeiro nodo da lista o *keeper* do arquivo (nodo *A*), e o restante *storers* (nodos *B, C* e *D*). O DIFS independe do protocolo e implementação da DHT, mas assume que falhas em nodos não interferirão na indexação dos arquivos. Por tratar-se de uma rede altamente heterogênea, onde podem existir nodos com quantidades variáveis de espaço em disco para compartilhar, o sistema, e não a DHT, deve decidir onde armazenar arquivos. Para isso, o nodo escolhido para armazenar o arquivo pela função *hash* da DHT armazena apenas o endereço do *keeper* e dos *storers* do arquivo. Desta forma, a DHT armazena apenas *links* para os arquivos, realizando a função de uma grande tabela FAT (*File Allocation Table*) distribuída.

Todos os *keepers* da rede devem checar periodicamente o estado dos seus respectivos *storers*. Sempre que um *keeper* detectar falha em algum *storer* ele criará uma nova réplica deste arquivo na rede. Falhas em *keepers* também podem ser detectadas, uma vez que os *storers* não mais receberão checagens periódicas. Caso isto ocorra, o próximo nodo da lista da DHT será o novo *keeper* do arquivo, criará uma nova réplica e atualizará a DHT.

2.1. Operações

O procedimento para ler, escrever, obter ou setar informações sobre arquivos é essencialmente o mesmo. Em todos esses casos, através da DHT é localizada a lista de nodos

que armazenam o arquivo. A partir dessa lista, a operação pode ser enviada a um nodo aleatório (operações de leitura) ou necessariamente ao keeper (operações de escrita). Um exemplo de operação é mostrado na figura 1. O nodo *A* deseja executar uma operação de leitura sobre o arquivo */arquivo*. Primeiramente, o nodo *A* busca na DHT os nodos que armazenam */arquivo*. O nodo *E* é o responsável por armazenar esta chave na DHT. A partir da resposta que o nodo *A* recebeu da DHT (que estava armazenada em *E*), ele conclui que o nodo *B* é o *keeper* deste arquivo e os nodos *D, F e C* os *storers*. O nodo *A* escolhe um dos nodos e envia diretamente a ele a requisição.

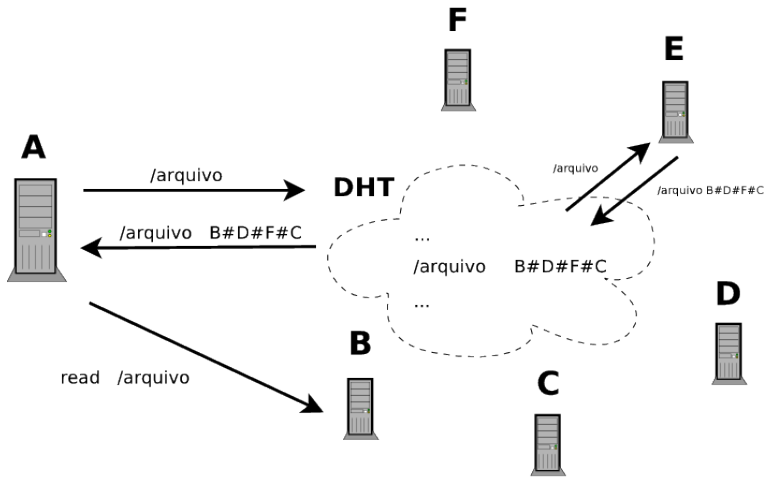

Figura 1. Exemplo de funcionamento do sistema.

3. Resultados Experimentais

Foi desenvolvido um protótipo do sistema na linguagem C, *open-source* e compatível com sistemas *Unix-like* com suporte ao FUSE. A partir deste protótipo foram realizados alguns experimentos, cujos resultados foram comparados aos do sistema de arquivos de rede NFS. Os testes de desempenho foram realizados utilizando 8 máquinas, com poderes computacionais diversos, conectadas por uma rede local (LAN) 100Mbps. Foi desenvolvido também um software que simula o comportamento de uma DHT. Apesar de ser somente um simulador, tal software é útil para estimar o *overhead* causado pela DHT no desempenho do sistema.

Para avaliar o desempenho do sistema foram realizados 4 testes. O primeiro teste tem o objetivo de avaliar o desempenho das operações de criação e escrita de arquivos da seguinte forma: são criados 2 arquivos de 512Kb com conteúdo aleatório e 1 diretório vazio para cada diretório existente no sistema. Todos os nodos executam este algoritmo uma vez, sequencialmente, de forma que, no final da execução do primeiro teste, um total de aproximadamente 512 arquivos de 512K espalhados em 128 diretórios estejam disponíveis na rede. Todos os arquivos são criados com conteúdo aleatório para possibilitar a verificação da correção de operações de leitura. O segundo teste mede o desempenho de operações de leitura de atributos sobre os arquivos criados pelo teste anterior. No terceiro teste, é medido o tempo gasto na leitura de todos os arquivos existentes no sistema. O quarto e último teste avalia o tempo necessário para leitura completa de um arquivo de 512Mb.

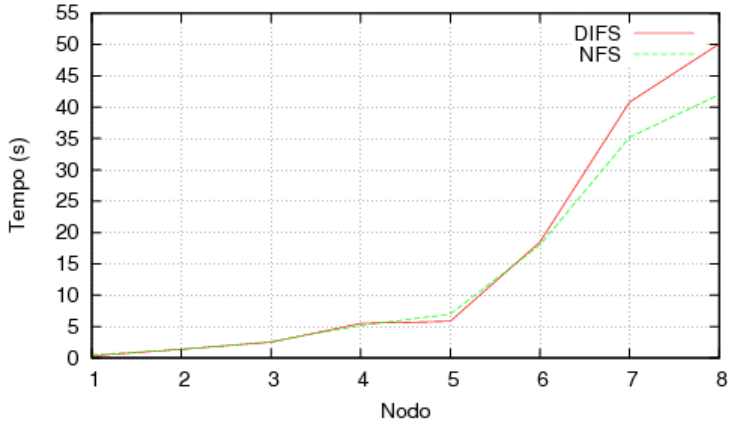

Figura 2. Resultados do teste 1.

No teste 1, cujos resultados são apresentados na figura 2, o número de arquivos e diretórios a serem criados aumenta a cada nodo. O tempo de execução do teste nos primeiros nodos é muito próximo, pois as máquinas passam grande parte do tempo gerando bits aleatórios para preencher os 512Kb dos arquivos. Conforme o número de arquivos a serem criados aumenta, e consequentemente a demanda por informações sobre as pastas e escritas no sistema, o NFS revela melhor desempenho, culminando em uma diferença de aproximadamente 20% na velocidade de execução do teste pelo último nodo, o nodo 8. Para os testes seguintes, apresentamos a média aritmética entre os tempos de execução de todos os nodos. A maior diferença de desempenho acontece no teste 2, onde o NFS mostra-se 4 vezes mais rápido que o sistema proposto (DIFS executou em 5.347s, o NFS em 1.364s). Uma das razões é que o NFS faz cache dos atributos dos arquivos, enquanto o protótipo do DIFS não implementa nenhum mecanismo de cache. Outro fator que contribui para a diferença de desempenho em todos os testes, é o *overhead* causado pelos acessos à DHT.

No teste 3, a diferença de desempenho cai para 2.8 (87.742s no DIFS e 30.971s no NFS). Esta diminuição pode ser atribuída ao fato de que, como a quantidade de dados a serem lidos é muito grande, o NFS não consegue mantê-los em cache. Situação semelhante ocorre no teste 4, onde o DIFS executou em 146.959s e o NFS em 59.067s, resultando em mais uma queda na diferença de desempenho: 2.5 vezes. A perda de desempenho neste teste é menor, pois como o DIFS não realiza particionamento de arquivos em blocos, após encontrado o nodo que armazena um arquivo, todas as operações de leitura são encaminhadas a ele.

4. Conclusão

Este trabalho apresentou o DIFS, um sistema de arquivos totalmente distribuído que utiliza uma DHT como rede overlay. Neste sistema, todos os nodos cedem uma certa quantidade de espaço para a rede, formando um grande sistema de arquivos compartilhado por todos. O DIFS utiliza o FUSE como interface com o kernel do sistema, possibilitando a sua utilização em modo usuário.

Os testes com o protótipo mostraram que o DIFS possui desempenho cerca de 3 vezes inferior ao NFS, comparando-se, em questão de desempenho, a sistemas de arquivos

distribuídos como Oceanstore e Ivy que possuem desempenho de 2 a 3 vezes inferior ao NFS. Entretanto, o protótipo ainda não implementa todas as funcionalidades do sistema e os testes constituem apenas uma análise preliminar de desempenho.

Um dos trabalhos futuros é a implementação de todas as funcionalidades descritas do sistema no protótipo e a realização de mais testes, em maior escala e simulando situações mais reais de uso. Além disso, estudos quanto ao número ideal de réplicas, tamanho de *read-ahead* e escolha do protocolo de DHT que minimize o tempo gasto na indexação, devem ser realizados. Outro trabalho interessante é um estudo sobre as caches no sistema, discutindo quais dados manter em cache, por quanto tempo e qual o tamanho máximo destinado à elas. O sistema também não implementa mecanismos de controle de consistência de réplicas ou controle de concorrência de escritas.

Referências

Carlsson, B. and Gustavsson, R. (2001). The rise and fall of Napster - an evolutionary approach. In *AMT '01: Proceedings of the 6th International Computer Science Conference on Active Media Technology*, páginas 347–354, London, UK. Springer-Verlag.

Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W. (2001). Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46–57.

eDonkey network. eDonkey protocol. `http://www.mirrorservice.org/sites/download.sourceforge.net/pub/sourceforge/p/pd/pdonkey/eDonkey-protocol-0.6.2.html`. Acessado no dia 26 de novembro de 2008.

FUSE. Fuse: Filesystem in userspace. `http://fuse.sourceforge.net/`. Acessado no dia 26 de janeiro de 2009.

Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., and Zhao, B. (2000). OceanStore: An architecture for global-scale persistent storage. In *Proceeedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*, páginas 190–201, Boston, MA.

Muthitacharoen, A., Morris, R., Gil, T., and Chen, B. (2002). Ivy: A read/write peer-to-peer file system. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI '02)*, Boston, Massachusetts.

OpenAFS. Openafs official web-site. `http://www.openafs.org/`. Acessado no dia 25 de novembro de 2008.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S. (2001). A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, páginas 161–172, New York, NY, USA. ACM.

Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, páginas 329–350.

Sandberg, R. (1986). The Sun Network File System: Design, implementation and experience. Technical report, in Proceedings of the Summer 1986 USENIX Technical Conference and Exhibition.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, páginas 149–160, San Diego, California.