CENTRO UNIVERSITÁRIO FEEVALE

LUCAS GRAEBIN

ARMAZENAMENTO DISTRIBUÍDO

Novo Hamburgo, Dezembro de 2006.

LUCAS GRAEBIN

ARMAZENAMENTO DISTRIBUÍDO

Centro Universitário Feevale

Instituto de Ciências Exatas e Tecnológicas

Curso de Ciência da Computação

Trabalho de Conclusão de Curso

Orientador: Reynaldo Cardoso Novaes

Novo Hamburgo, Dezembro de 2006.

RESUMO

A constante onipresença das redes proporcionou um meio de comunicação comum entre grande parte dos usuários de microcomputadores. Este canal de transmissão facilitou o processo de armazenamento e compartilhamento de arquivos entre usuários através de serviços como o *Peer-to-Peer* (P2P) de forma mais segura em relação aos dispositivos móveis de pouca confiabilidade utilizados até então. Empresas de grande porte como a Microsoft e a IBM, além de Universidades, vêm investindo na pesquisa e no desenvolvimento de ferramentas que objetivam o armazenamento distribuído de arquivos. Estes projetos, individualmente, são direcionados à resolução de problemas específicos, que vão desde o reaproveitamento de espaço em disco ocioso até a possibilidade de o cliente trabalhar com seus arquivos tendo sua estação de trabalho desconectada da rede. Um sistema de armazenamento distribuído é uma abstração de acesso aos discos rígidos de microcomputadores dispersos fisicamente pela rede. Uma rede de armazenamento distribuído é composta de clientes e servidores, onde as estações podem desempenhar ambos os papéis. Sua implementação traz como obstáculo os principais desafios de sistemas distribuídos, como a transferência, a replicação, a segurança, a concorrência etc. Este trabalho aborda conceitos e ferramentas para o armazenamento distribuído de arquivos, onde serão tratadas as técnicas empregadas por determinadas soluções para resolver os problemas de replicação e transparência de acesso e transparência de localização.

Palavras-chave: Sistemas Distribuídos, Sistemas de Arquivos Distribuídos, Armazenamento Distribuído, *Peer-to-Peer*.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AFS	Andrew *File System*
CORBA	*Common Object Request Broker Architecture*
DFS	*Distributed File System*
GFS	Google *File System*
ISO	*International Organization for Standardization*
NFS	*Network File System*
P2P	*Peer-to-Peer*
RPC	*Remote Procedure Call*
URL	*Universal Resource Locator*
USB	*Universal Serial Bus*
VFS	*Virtual File System*
XDR	*External Data Representation*

SUMÁRIO

INTRODUÇÃO

Por muito tempo, os discos flexíveis eram os únicos meios para o compartilhamento e transferência de dados entre usuários de computadores pessoais. Embora seu uso fosse trivial, a disciplina usual era de possuir uma segunda cópia do disco para assegurar a disponibilidade dos dados e evitar perdas. Devido a sua instabilidade, baixa capacidade e taxa de transferência, os discos flexíveis passaram a ser vistos como uma alternativa pouco confiável. Com o passar do tempo, surgiram alternativas de armazenamento mais confiáveis, tais como o CD. Apesar de poder ser lido em grande parte dos microcomputadores, a manipulação dos dados ali armazenados era inconveniente, pois dependia de um dispositivo de gravação não presente em todos os equipamentos. Com o advento das tecnologias *Universal Serial Bus* (USB) e IEEE-1394 (também conhecido como *Firewire*), o mercado de mídia removível foi revitalizado. Entretanto, apesar de sua robustez e elevado grau de confiabilidade em relação aos discos flexíveis, estes dispositivos não garantem a total disponibilidade dos dados, sendo vulneráveis a problemas eletrônicos, possíveis furtos e perdas do equipamento. Devido a estas inseguranças, os usuários tiveram que aderir a serviços disponíveis na Internet para o armazenamento e transferência de dados.

Com a crescente oferta de largura de banda e a redução acentuada de seu custo para utilização dos meios de transmissão, os protocolos HTTP, FTP e serviços *Peer-to-Peer* (P2P) (DABEK et al., 2001) tornaram-se alternativas de baixo custo para o armazenamento e transferência de arquivos entre computadores pessoais. Apesar destas técnicas, estes ambientes, em sua maioria, não garantem a disponibilidade dos dados, sendo vulneráveis a problemas de hardware e software. Alguns analistas prevêem que, no futuro, os dados que hoje são armazenados localmente, residirão em servidores existentes na Internet (ROUSH, 2006). Diversas empresas oferecem serviços que exploram esta forma de armazenamento, como por exemplo, o BeInSync (BEINSYNC, 2006) e o MediaMax da Streamload (MEDIAMAX, 2006). Nestes casos, os problemas de disponibilidade são tratados com o uso

de uma infra-estrutura própria da empresa. Além dos exemplos acima, empresas de grande porte, como a Microsoft, oferecem serviços como o FolderShare (FOLDERSHARE, 2006) e possuem pesquisas nesta área (PAST, 2006) o que demonstra o potencial deste tipo de solução em um futuro próximo.

Apesar de existirem soluções proprietárias para o repositório de arquivos, diversos esforços baseados em Software Livre e com iniciativas em outras direções, como por exemplo o reaproveitamento de espaços em disco ociosos entre os computadores de uma rede, vêm sendo direcionados para esta área, a exemplo do LoDN (LODN, 2006), o OpenAFS (OPENAFS, 2006) e o Coda (CODA, 2006).

Para ser bem sucedido, o armazenamento distribuído deve tratar um dos grandes desafios dos sistemas distribuídos que é a transparência. De acordo com Coulouris, a transparência é definida como “o encobrimento do usuário e do programador de aplicação da separação dos componentes de um sistema distribuído, de modo que o sistema seja percebido como um todo ao invés de uma coleção de componentes independentes” (COULOURIS, 2005, p. 23) (TRADUÇÃO NOSSA). Para o armazenamento distribuído dos arquivos, a transparência é obtida oferecendo ao usuário um modelo de armazenamento que estende o tradicional conceito de armazenamento presente nos computadores domésticos.

Um sistema de arquivos distribuído “é a implementação de um sistema de arquivos que consiste em locais de armazenamento fisicamente dispersos, mas que provê uma visão centralizada tradicional de sistema de arquivos pelos usuários” (CHOW, 1998, p. 192) (TRADUÇÃO NOSSA). Silberschatz complementa que o sistema de arquivos distribuído é composto de clientes, servidores e dispositivos de armazenamento dispersos na rede, onde não há um único repositório de arquivos, mas sim diversos, sendo eles independentes (SILBERSCHATZ, 2000). Dessa forma, pode-se resumir que os sistemas de armazenamento distribuído constituem de uma camada de abstração para a realização de tarefas como gravação e leitura em discos rígidos espalhados fisicamente na rede.

Os mais importantes desafios na construção de sistemas distribuídos estão presentes quando se implementa sistemas de armazenamento distribuído e devem ser levados em conta de acordo com o ambiente a que se destinam. Estes aspectos são a já citada transparência, resolução de nomes, gerenciamento de replicações e segurança (CHOW, 1998). O grau de importância destes desafios na implementação dos sistemas de arquivos distribuídos, além de

depender do ambiente, recebem tratamentos diferentes entre os autores da área de sistemas distribuídos. De acordo com Singhal, os dois serviços de maior importância presentes nos sistemas de armazenamento distribuído são o servidor de nomes e o gerenciador de *cache* (SINGHAL, 1994). Estes serviços são, respectivamente, responsáveis pelo mapeamento de arquivos e diretórios, e pelo armazenamento de cópias dos dados de usuários em seus computadores (SINGHAL, 1994). Chow, por sua vez, ressalta a importância da dispersão e da multiplicidade, onde múltiplos clientes dispersos fisicamente podem acessar diversos arquivos residentes em servidores também espalhados em diversas localizações (CHOW, 1998).

A diversidade de abordagens em trabalhos teóricos, como citado acima, também tem reflexos nas implementações existentes atualmente. O resultado disso pode ser observado na implementação de diversos sistemas para armazenamento distribuídos que vêm sendo desenvolvidos e mantidos nos últimos anos, cada um deles apresentando a resolução de um problema específico.

Este trabalho tem por objetivo fazer uma revisão bibliográfica dos conceitos, definições, desafios e problemas gerados com a implementação de sistemas distribuídos e sistemas para armazenamento distribuído. Na segunda parte deste trabalho, serão realizados estudos e experimentos quanto as técnicas implementadas por determinadas ferramentas de armazenamento distribuído para solucionar os desafios de transparência de acesso, transparência de localização e replicação, com base no cenário proposto nesta primeira etapa.

Além desta introdução, o trabalho esta organizado em mais cinco capítulos. No primeiro capítulo, será feita uma revisão nos conceitos, definições, desafios e problemas existentes no desenvolvimento de sistemas distribuídos. No segundo capítulo, será feita uma revisão na área de armazenamento distribuído, abrangendo seus conceitos, definições, desafios e problemas. No capítulo três, são apresentadas algumas ferramentas que fornecem o serviço de armazenamento distribuído e como elas resolvem, ou não, os desafios de transparência e replicação. No quarto capítulo, será definido o cenário a ser utilizado na segunda etapa deste trabalho para a realização dos experimentos propostos. Por fim, são apresentadas as considerações finais.

1 SISTEMAS DISTRIBUÍDOS

As redes estão presentes em todos os lugares. As primeiras evidencias de redes de computadores apareceram no início da década de 60 com o projeto e implementação de sistemas centralizados, contendo um grande número de terminais espalhados (KIRNER, 1988). A partir de então, a evolução tecnológica, possibilitando uma redução nos custos de comunicação e de *hardware*, aliada ao avanço no estudo de redes, propiciou as condições favoráveis ao surgimento das redes de comunicação de computadores, proporcionando o compartilhamento de recursos entre máquinas e o aumento da confiabilidade e desempenho em redes locais (KIRNER, 1988). Inicialmente, foi dada ênfase a problemas simples, como a troca de mensagens entre usuários de diferentes máquinas e o compartilhamento de impressoras ou de discos rígidos apenas para leitura, entretanto, com o passar do tempo, foram desenvolvidos sistemas mais complexos que oferecessem ao usuário um maior aproveitamento dos recursos distribuídos pela rede (KON, 1994).

O compartilhamento de recursos computacionais pode existir de forma explícita ou implícita (KIRNER, 1988). No compartilhamento explícito, a rede oferece a possibilidade dos usuários acessarem e manipularem diretamente os recursos remotos, exigindo que os clientes conheçam os detalhes dos recursos, enquanto que no compartilhamento implícito, o acesso é feito automaticamente pelo sistema de forma transparente para o usuário. Sistemas com compartilhamento de recursos de comunicação e compartilhamento implícito de recursos acabaram, em grande parte, constituindo-se nos sistemas distribuídos (KIRNER, 1988).

Várias definições para sistemas distribuídos podem ser encontradas na literatura. De acordo com Coulouris, um sistema distribuído é um “sistema cujos componentes, *software* ou *hardware*, localizados em computadores conectados a uma rede, comunicam-se e coordenam suas ações unicamente através da transmissão de mensagens” (COULOURIS, 2005, p. 2) (TRADUÇÃO NOSSA). Tanenbaum define sistemas distribuídos como sendo “[...] uma

coleção de computadores independentes que aparecem para os usuários do sistema como um único computador" (TANENBAUM, 2002, p. 2) (TRADUÇÃO NOSSA). Ribeiro complementa apresentando uma definição mais detalhada:

> "Os sistemas distribuídos foram criados para distribuir as tarefas e aumentar o poder computacional através do uso de vários processadores como também promover o compartilhamento de recursos. Cada processador possui sua própria memória e a comunicação entre os processadores é feita através de linhas de comunicação. Esses sistemas objetivam melhorar a comunicação entre os computadores, propiciando a integração destes num sentido amplo, que pode envolver a facilidade de mudanças futuras, rapidez nas trocas de informações e confiabilidade na execução dos processos. Eles permitem que uma aplicação seja dividida em diferentes partes que podem ser processadas em sistemas independentes que se comunicam através das linhas de comunicação. Ele cria a ilusão de que a rede de computadores seja vista como um único sistema de tempo compartilhado, em vez de um conjunto de máquinas distintas." (RIBEIRO, 2005, p. 32).

As diferenciações na definição de sistemas distribuídos e a conseqüente falta de clareza de algumas têm gerado controvérsias no que diz respeito à relação entre sistemas distribuídos e as redes de computadores (TANENBAUM, 2003). "A distinção fundamental é que em um sistema distribuído, uma coleção de computadores independentes mostra-se aos usuários como sendo um sistema único" (TANENBAUM, 2003, p. 2) (TRADUÇÃO NOSSA). De acordo com Verissimo, uma rede de computadores é uma infra-estrutura que conecta máquinas e adota um conjunto de protocolos de comunicação para permitir a interação entre elas, enquanto que os sistemas distribuídos são construídos sobre as redes de computadores, que hospeda serviços nas máquinas da rede e faz uso de protocolos de comunicação para suportar a execução coerente das aplicações (VERISSIMO, 2001).

Os sistemas distribuídos devem possuir, obrigatoriamente, características de multiplicidade de nós de processamento, para permitir o aumento no desempenho, tolerância a falhas e disponibilidade do sistema, e um mecanismo de comunicação para suportar a conversação entre os componentes do sistema distribuído (SPECTOR, 1981). Excepcionalmente, características como, a possibilidade de expansão, para permitir o crescimento incremental do sistema, mecanismos para detecção de erros e redundância de dispositivos, para obter disponibilidade e tolerância a falhas, e dispersão geográfica, para permitir que os recursos sejam separados geograficamente, devem ser agregadas em seu desenvolvimento (SPECTOR, 1981).

Quando bem implementado, um sistema distribuído pode oferecer diversas vantagens em relação aos sistemas centralizados e sobre um conjunto de máquinas isoladas, tais como o

compartilhamento de recursos, aumento da disponibilidade dos serviços, possibilidade de expansão da arquitetura etc (KON, 1994; COULOURIS, 2005).

1.1 Desafios de Sistemas Distribuídos

O projeto de um sistema distribuído é geralmente mais complexo em relação aos sistemas centralizados, pois seus recursos estão, muitas vezes, fisicamente dispersos, não havendo nenhum relógio comum entre os múltiplos processadores, podendo ocorrer atrasos na entrega das mensagens ou estas podem, até mesmo, serem perdidas. Apesar destas complexidades e dificuldades, os usuários devem ver um ambiente de sistema distribuído como sendo um sistema centralizado virtual, que oferece características como flexibilidade, eficiência, segurança e facilidade em seu uso (SINHA, 1996). Esta seção tem por objetivo apresentar os principais conceitos e desafios existentes na implementação de sistemas distribuídos.

1.1.1 Heterogeneidade

A heterogeneidade pode ocorrer de várias formas nos sistemas distribuídos, variando desde a diversidade de *hardware* e diferenças em protocolos de interligação de redes até variações em gerenciadores de dados. Devido a estas distinções, a integração entre sistemas em ambientes heterogêneos é mais complexa em relação a ambientes homogêneos, onde cada sistema é baseado em uma arquitetura similar ou equivalente de *hardware* e *software* (NOTKIN, 1987). A Internet é um exemplo prático de ambiente heterogêneo, pois concede aos usuários o acesso a serviços e a execução de aplicações sobre um conjunto de computadores e redes de comunicação com características diferenciadas (COULOURIS, 2005).

A heterogeneidade é freqüentemente inevitável, podendo ser originada quando surgem necessidades de aquisição e desenvolvimento de sistemas ou equipamentos de *hardware* (NOTKIN, 1987). Ela pode fazer-se presente de diferentes formas em um sistema distribuído (COULOURIS, 2005):

Rede: Embora a Internet seja composta por variações de tipos de redes, suas diferenças são mascaradas pelo fato de todos os computadores utilizarem um protocolo em comum para a comunicação.

Hardware: Tipos de dados, como *integers*, podem ser representados de diferentes formas dependendo da arquitetura de *hardware*. Nestes casos, as diferenças de representação devem ser negociadas nas mensagens que serão trocadas entre os programas.

Sistemas Operacionais: Embora todos os sistemas operacionais de computadores conectados na Internet necessitem de uma implementação de um protocolo de comunicação em comum, estes possuem interfaces e chamadas de sistemas diferentes entre si. Por exemplo, a chamada para troca de mensagens no Unix é diferente da chamada do Windows.

Linguagens de Programação: As linguagens de programação utilizam distintas representações de estrutura de dados como *array* e registros. Estas diferenças devem ser endereçadas caso programas escritos em linguagens diferentes comuniquem-se entre si.

Implementação por diferentes programadores: Programas escritos por diferentes programadores não podem comunicar-se entre si, a menos que eles utilizem padrões comuns, como uma mesma estrutura para a troca de mensagens pela rede.

Não há uma única solução correta para se resolver os problemas de ambientes heterogêneos. Para suportar esta dificuldade, os sistemas distribuídos são compostos geralmente de uma camada de *software* denominada *middleware*, que é inserida logicamente entre os usuários e o nível mais elevado da aplicação para fornecer uma abstração quanto ao protocolo de comunicação, sistema operacional e *hardware* (TANENBAUM, 2002), conforme ilustrado na Figura 1.1.

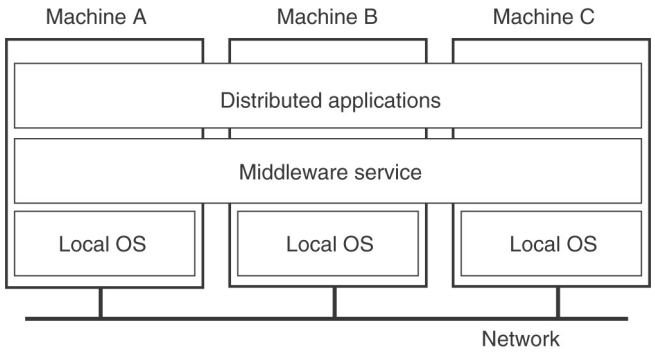

Figura 1.1 - Exemplo de um sistema distribuído composto por uma camada de *middleware*
Fonte: Tanenbaum, 2002, p. 3

1.1.2 Tolerância a Falhas

Sistemas computacionais não estão livres de falhas. Quando elas ocorrem no nível de *hardware* ou *software*, os programas podem produzir resultados incorretos, ou podem ser interrompidos antes da conclusão de suas operações (COULOURIS, 2005). Um sistema é considerado tolerante a falhas quando este consegue mascarar a presença de falhas através das técnicas de redundância (JALOTE, 1994).

Para que um sistema seja confiável, ele deverá continuar funcionando corretamente mesmo com a presença de certos tipos de erros ou interferências indevidas (KIRNER, 1988). Confiança é um termo que abrange várias exigências úteis para sistemas distribuídos, como disponibilidade, confiança, segurança e sustentabilidade (MULLENDER, 1993). A disponibilidade corresponde a probabilidade de o sistema manter-se operacional dentro de certo período de tempo. A confiabilidade é o grau de sucesso que um sistema consegue atingir dentro de um período de tempo, mantendo seu comportamento dentro das especificações normais. Em contraste com a disponibilidade, a confiança é definida em termos de intervalo de tempo ao invés de um determinado momento, que é quando ocorreria a chamada do sistema pelo usuário. A segurança refere-se a situação em que nenhum efeito catastrófico venha a ocorrer originário de uma falha temporária do sistema, e a sustentabilidade, por sua vez, refere-se a possibilidade do sistema detectar fracassos e consertá-los automaticamente (MULLENDER, 1993).

Uma das maneiras para se classificar as falhas em sistemas distribuídos é baseada no comportamento do componente após o ocorrido (JALOTE, 1994). As falhas podem ocorrer a nível de processo ou canal de comunicação e são classificadas em (COULOURIS, 2005; JALOTE, 1994):

Omissão: Recorrem a casos em que um processo ou um canal de comunicação não responde a pedidos entrantes.

Arbitrária (ou Bizantinas): Nestes casos, o componente comporta-se de maneira totalmente arbitrária durante a falha. Por exemplo, um processo pode ajustar os valores errados a uma variável, ou pode retornar um valor errado como resposta a uma requisição. As falhas arbitrárias não podem ser descobertas apenas vendo se o processo responde as

requisições, pois poderia-se arbitrariamente omitir a resposta correta retornando um valor errado como resposta a uma solicitação.

Temporização: Esta falha ocorre quando um componente responde demasiadamente cedo ou demasiadamente tarde. A temporização é bastante relevante em sistemas multimídia, onde há a transferência de áudio e vídeo. Em sistemas assíncronos, pode haver uma alta latência entre o pedido do cliente e a resposta do servidor caso este esteja sobrecarregado, entretanto não se pode afirmar que ocorreram fracassos na solicitação, pois este modelo de sistema não oferece garantia de tempo de resposta.

Em sistemas tolerantes a falhas, a existência de defeitos e interferências indevidas pode ser superada através de redundância temporal, redundância de *hardware* ou redundância de *software*, técnicas geralmente utilizadas na construção de sistemas distribuídos (JALOTE, 1994). A redundância temporal corresponde a determinação de um resultado através de execuções repetidas da mesma operação pelo mesmo elemento, usando eventualmente métodos diferentes. A redundância de *hardware* compreende do acréscimo de elementos repetitivos capazes de executarem a mesma operação. A redundância de *software* inclui todos os programas e instruções que são empregadas para dar suporte à tolerância a falhas (JALOTE, 1994).

As partes redundantes do sistema podem ser utilizadas como elementos de votação ou como elementos de reserva. Atuando como elementos de votação, cada parte redundante permanecerá ativa no sistema, fornecendo seu resultado que, comparado com os outros, contribuirá para a escolha do resultado correto por maioria. Como elemento de reserva, cada parte redundante permanecerá latente até que seja acionada pelo sistema para substituir outro elemento defeituoso que estiver sendo desativado. Neste último caso, o sistema deverá possuir um bom mecanismo para a realização de testes e maneiras eficientes de identificação de defeitos para poder localizá-los e superá-los (KIRNER, 1988).

Os sistemas tolerantes a falhas situam-se em duas classes: sistemas de alta disponibilidade e sistemas de alta confiabilidade. Enquanto que sistemas de alta disponibilidade podem tolerar pequenos atrasos decorrentes de manutenção, os sistemas de alta confiabilidade exigem que o funcionamento correto seja mantido, apesar da presença de faltas (KIRNER, 1988).

O objetivo principal da abordagem tolerante a falhas consiste em inibir o efeito das faltas através das redundâncias do sistema. Quando estes sistemas forem submetidos à ocorrência de uma falha, eles deverão reagir executando alguns, ou todos os procedimentos a seguir (SIEWIOREK, 1984):

Confinar a falta: Consiste em limitar o alcance de seus efeitos través do uso de circuitos de detecção de faltas, verificação de consistência e precedendo certas operações, protocolos múltiplos de comunicação etc., evitando dessa forma a propagação das falhas.

Detectar a falta: Corresponde a tomar conhecimento de sua existência através de *bits* de paridade, verificação de consistência, violação de protocolo etc. Essa detecção poderá ser feita com o dispositivo fora de operação ou em operação.

Mascarar a falta: Tem por objetivo eliminar os seus efeitos, fazendo com que as informações redundantes se imponham sobre as informações incorretas.

Tentar novamente: Este é um procedimento que dará bons resultados quando se tratar de faltas transientes, que não provoquem nenhum dano físico ao sistema, podendo ser bem sucedida em uma segunda tentativa.

Diagnosticar: Esta etapa tem por objetivo identificar a localização da falta e suas propriedades, uma vez que a detecção da falta não trata desse aspecto.

Re-configurar: Consiste em substituir ou isolar as partes defeituosas. A alternativa por isolar as partes defeituosas poderá levar o sistema a uma degradação gradual. A substituição das partes defeituosas só será possível se houverem disponíveis partes correspondentes.

Recuperar: Tem a função de colocar a operação do sistema, ou de alguma de suas partes, numa situação anterior a ocorrência das falhas com o objetivo de eliminar os erros que aparecem. Isto exigirá do sistema, conhecimento de pontos de operação que possam ser restaurados. Algumas técnicas utilizadas para recuperar pontos de operação anteriores consistem em manter arquivos de *backup*, usar pontos de verificação, manipular listas de intenção etc.

Reiniciar: Consiste em recolocar o sistema em funcionamento. Este processo poderá não ser realizado totalmente com sucesso caso o sistema tenha sido danificado gravemente

pelo erro, ou este não possua suporte para este procedimento. O reinício “quente” corresponde a retomada de todas as operações a partir do ponto de onde ocorreu a detecção da falta. O reinício “morno” ocorrerá quando somente uma parcela do sistema estiver em condições de ser retomada. O reinício “frio” consistirá no reinício total do sistema, precedido pela recarga, se for necessária, para superar perdas de grande amplitude.

Reparar: Constitui-se na atividade de consertar o componente defeituoso. Isto poderá ser realizado com o sistema fora de operação ou em operação. No caso de reparo com o sistema fora de operação, o elemento defeituoso deverá ser consertado com o sistema desligado. No caso de reparo com o sistema em operação, o elemento defeituoso deverá ser substituído por um elemento sobressalente nos moldes da re-configuração, ou deverá ser retirado, ocasionando eventualmente uma degradação gradual no sistema.

Reintegrar: Após a reparação do elemento, este deverá ser reintegrado no sistema. Para sistemas em operação, a reintegração deverá ser efetuada sem a interrupção das operações.

1.1.3 Segurança

A finalidade da camada de segurança em sistemas distribuídos é de restringir o acesso a informação e aos recursos para usuários e processos não autorizados, fortalecendo a privacidade de indivíduos e organizações. Muitos dos recursos existentes nos sistemas distribuídos lidam com informações de elevado valor intrínseco para os usuários (COULOURIS, 2005), o que justifica a importância da segurança nestes sistemas.

Implementar técnicas de segurança em sistemas distribuídos é mais complexo em relação aos sistemas centralizados devido a falta de um único ponto de controle e do uso de redes inseguras para a transmissão de dados (SINHA, 1996).

Tanenbaum (2002) diz que a segurança em sistemas computacionais esta relacionada fortemente com a noção de confiança dos usuários, que usufruirão dos serviços. Os serviços de segurança caracterizam os diferentes aspectos de um sistema de computador, tais como (STALLINGS, 2003):

Autenticidade: Requer que a origem ou o originador de uma mensagem seja corretamente identificado. A verificação da autenticidade é necessária após todo processo de

identificação, seja de um usuário para um sistema, de um sistema para o usuário ou de um sistema para outro sistema.

Integridade: Consiste em proteger a informação contra modificação sem a permissão explicita do proprietário daquele dado. A modificação inclui ações como escrita, alteração de conteúdo, alteração de *status*, remoção, criação e o atraso de informações transmitidas.

Confidencialidade: Consiste em proteger a informação contra leitura ou cópia por alguém que não tenha sido explicitamente autorizado pelo proprietário. Este tipo de segurança inclui não apenas a proteção da informação como um todo, mas também de fragmentos dela, que podem ser utilizadas para inferir sobre o todo.

Controle de Acesso: Refere-se a capacidade de se permitir ou negar acesso aos serviços e recursos oferecidos pelo sistema. Acessos desconhecidos ou feitos por pessoas não autorizadas podem significar a necessidade de uma verificação de todos os recursos envolvidos em busca de possíveis estragos que possam ter sido causados ao sistema, mesmo que aparentemente nada tenha ocorrido.

Para proteger os serviços e informações contidas nos sistemas computacionais, deve-se ficar atento às ameaças de segurança, como ataques de negação de serviço, que podem gerar perdas ou dados à informação. De acordo com Tanenbaum, há quatro tipos de ameaças de segurança que se deve considerar (TANENBAUM, 2002):

Intercepção: Ocorre quando um componente do sistema é acessado sem autorização.

Interrupção: Refere-se a situação em que o componente do sistema é corrompido ou torna-se indisponível, como por exemplo, um rompimento de uma linha de comunicação.

Modificação: Envolve a alteração de um componente do sistema por parte de um usuário não autorizado, como por exemplo, a modificação do valor de pagamento de uma compra via Internet durante o trânsito das informações pela rede.

Fabricação: Refere-se a situações onde são inseridos objetos falsos em um componente do sistema.

Com o intuito de evitar essas ameaças, Coulouris (2005) e Tanenbaum (2002) mencionam alguns mecanismos e técnicas para a segurança em sistemas distribuídos e aplicativos, tais como:

Criptografia: A criptografia é o processo de codificar uma mensagem fazendo-se uso de algoritmos e chaves de segurança. A chave criptográfica é um parâmetro utilizado nos algoritmos de criptografia que garante que ela possa ser invertida sem o conhecimento da chave. Existem dois métodos de criptografia. No primeiro, denominado criptografia simétrica, o emissor e o receptor compartilham uma chave secreta que não pode ser de conhecimento de terceiros. Esta chave é utilizar para criptografar e descriptografar a mensagem. Na segunda técnica, denominada de criptografia assimétrica, são utilizados pares de chaves para criptografar e decifrar as mensagens. O emissor possui uma chave pública do destinatário e este faz uso dela para criptografar a mensagem antes de enviá-la. O destinatário utiliza a chave privada correspondente para decifrar a mensagem.

Autenticação: A autenticação é o mecanismo utilizado para garantir a real identidade de um usuário ou processo. A autenticação pode ser feita de forma unilateral, mútua ou com a mediação de terceiros, dependendo da situação. A autenticação unilateral acontece quando apenas um processo da comunicação autentica-se perante o outro, mas a recíproca não é verdadeira. Na autenticação mútua os processos envolvidos na comunicação se autenticam um perante o outro. A autenticação com a mediação de terceiros é utilizada quando os processos da comunicação não se conhecem e dessa forma não podem se autenticar mutuamente, mas conhecem um terceiro com quem se autenticam e recebem credenciais para procederem assim a autenticação mútua.

Controle de Acesso: As técnicas de controle de acesso propõem-se a definir quais usuários ou processos possuirão acesso a um determinado recurso do sistema computacional e quais as permissões que a ele serão concedidas. Estas técnicas podem ser aplicadas em qualquer nível, desde a aplicação, definindo quais usuários possuirão acesso a determinados registros, até ao sistema básico, determinando quais processos possuirão acesso a determinadas páginas de memória.

1.1.4 Concorrência

Serviços e aplicações provêem recursos que podem ser compartilhados e solicitados simultaneamente pelos clientes em sistemas distribuídos (COULOURIS, 2005). Sempre que existir compartilhamento e acesso simultâneo, deve-se assegurar que as requisições sejam processadas em sua ordem de chegada para evitar que resultados inexatos venham a ocorrer (GALLI, 2000). Entretanto, em um sistema distribuído, às vezes é impossível determinar a ordem exata em que dois eventos ocorreram, uma vez que estes ambientes não possuem memória e nem *clock* em comum (SILBERSCHATZ, 2000).

Uma das soluções para evitar os problemas de concorrência é impedindo que dois ou mais processos acessem um mesmo recurso simultaneamente. A exclusão mútua garante que, enquanto um processo estiver acessando determinado recurso, todos os demais processos interessados no uso deste recurso deverão aguardar pelo término de sua utilização. A parte do código do programa que acessa o recurso compartilhado é denominada de região crítica. Somente tal região necessita proibir o acesso concorrente. O controle da concorrência nesta região pode ser gerenciado com a implementação e uso de semáforos e monitores (GALLI, 2000).

O semáforo é uma variável do tipo *integer* cujo seu valor indica o estado do recurso protegido, que só pode ser manipulada por duas únicas instruções: P e V (GALLI, 2000). Estas denominações são as iniciais das palavras holandesas *Proberen* e *Verhogen*, que significam testar e incrementar, respectivamente (MACHADO, 2002). O valor do semáforo igual a 1, indica que nenhum processo está utilizando o recurso, enquanto que o valor igual a 0, indica que o recurso está em uso. Sempre que um processo desejar entrar em sua região crítica, a instrução P é executada, e caso o semáforo seja igual a 1, este valor será decrementado e o processo solicitante poderá executar as instruções de sua região crítica. Entretanto, caso uma instrução P seja executada em um semáforo com valor igual a 0, o processo ficará impedido de possuir o acesso, permanecendo em estado de espera (TANENBAM, 2006; MACHADO, 2002; GALLI 2000).

O monitor representa um conjunto de variáveis, de procedimentos e de estruturas de dados que são agrupados em um tipo especial de módulo ou de pacote. Os processos podem invocar os procedimentos de um monitor sempre que quiserem, mas eles não podem acessar diretamente as estruturas de dados internas do monitor a partir de procedimentos declarados

fora dele. Em um monitor, apenas um processo pode estar ativo em qualquer momento, o que torna esta técnica útil para obter a exclusão mútua. Quando um processo chama um procedimento do monitor, as primeiras instruções do procedimento verificarão se existem processos ativos. Caso exista, o processo de chamada será suspenso até que o outro processo tenha deixado o monitor. Se nenhum outro processo estiver utilizando o monitor, o processo de chamada pode entrar (TANENBAUM, 2006; GALLI, 2000).

1.1.5 Transparência

A transparência é definida como “o encobrimento do usuário e do programador da aplicação da separação dos componentes de um sistema distribuído, de modo que o sistema seja percebido como um todo ao invés de uma coleção de componentes independentes” (COULOURIS, 2005, p. 23) (TRADUÇÃO NOSSA). Em outras palavras, a transparência tem por objetivo esconder, sempre que possível, todos os detalhes do sistema que são irrelevantes do usuário (CHOW, 1998). Alcançar a transparência completa é uma tarefa complexa e requer que diversos aspectos, identificados pela *International Organization for Standardization* (ISO), sejam levados em consideração durante o desenvolvimento (COULOURIS, 2005; TANENBAUM, 2002):

Transparência de Acesso: Permite que o acesso a recursos locais e recursos remotos seja feito através das mesmas operações.

Transparência de Localização: Permite que os recursos possam ser acessados sem o conhecimento de sua real localização. A transparência de localização possibilita que recursos sejam movidos entre nodos de um sistema distribuído, como por exemplo, um arquivo, sem ter seu nome e caminho afetado.

Transparência de Concorrência: Permite que os diversos processos possam operar concorrentemente usando um recurso compartilhado sem que cause interferência aos demais processos que estão utilizando-o.

Transparência de Replicação: Permite que os usuários façam uso das múltiplas instancias dos recursos com o intuito de aumentar a confiabilidade e o desempenho sem que estes saibam que estão manuseando réplicas.

Transparência de Falha: Permite o encobrimento das falhas de *software* e *hardware*, possibilitando aos usuários a conclusão de suas tarefas sem que tomem conhecimento dos possíveis fracassos do sistema.

Transparência de Migração: Permite o movimento de recursos e clientes dentro de um sistema sem afetar a operação do usuário ou da aplicação e sem que estes tomem conhecimento desta operação.

Transparência de Desempenho: Permite a re-configuração do sistema para melhorar seu desempenho sem que os usuários percebam.

Transparência de Escalabilidade: Permite que o sistema seja expandido sem que haja a necessidade de alterar sua estrutura ou os algoritmos das aplicações.

As técnicas de transparência de maior importância são a de acesso e localização, pois a presença ou ausência delas afeta fortemente a utilização dos recursos (COULOURIS, 2005). Como exemplo de uma transparência de acesso, pode-se considerar uma interface onde o usuário possa acessar seus arquivos e diretórios locais e remotos, enquanto que um exemplo da falta da transparência de acesso seria um sistema distribuído que não permitisse o acesso aos arquivos em um computador remoto a menos que o cliente faça uso, por exemplo, de um aplicativo de FTP (COULOURIS, 2005).

A transparência de localização pode ser exemplificada através da URL de um determinado servidor *Web*, onde para se ter acesso a informação, o cliente não precisa ter conhecimento da real localização da máquina (COULOURIS, 2005).

1.1.6 Escalabilidade

Um sistema é denominado escalável quando este for projetado para lidar facilmente com o aumento de recursos e usuários, como a exemplo da Internet, que possui um crescimento dramaticamente alto de computadores e serviços (COULOURIS, 2005). Sinha (1996) menciona alguns dos princípios utilizados para projetar sistemas escaláveis:

Evitar sistemas centralizados: O uso de sistemas centralizados, como um único servidor de arquivo ou um único banco de dados, não é considerado um sistema escalável, podendo apresentar perda de desempenho e tornar-se um gargalo quando houverem aumentos demasiados de requisições para ele.

Evitar algoritmos centralizados: Um algoritmo centralizado é aquele que opera em um único servidor, coletando informações dos diversos nodos da rede e redistribuindo-as a eles. O uso de tais algoritmos no projeto de sistemas distribuídos também não é considerado como escalável, pelo mesmo motivo dos sistemas centralizados.

Execute a grande parte das operações em estações de trabalho de clientes: Quando possível, determinadas operações deveriam ser executadas no cliente ao invés do servidor, pois ele é um recurso comum para os demais membros da rede e conseqüentemente, os ciclos de um servidor são mais custosos em relação aos ciclos de uma estação de trabalho. Este princípio realça a escalabilidade do sistema e reduz a disputa por recursos compartilhados.

2 SISTEMAS DE ARQUIVOS DISTRIBUÍDOS

Os sistemas de arquivos foram originalmente desenvolvidos para sistemas computacionais centralizados e computadores *desktop* como uma facilidade do sistema operacional para prover uma interface conveniente para o armazenamento de arquivos em disco (COULOURIS, 2005). Subseqüentemente, eles adquiriram recursos como a possibilidade de delegar permissão de acesso sobre os arquivos e diretórios, que passou a ser útil quando se necessitava compartilhá-los. Com o passar dos tempos, surgiu a necessidade do compartilhamento de arquivos entre usuários de uma mesma rede, dando origem aos tradicionais servidores de arquivos. Um servidor de arquivos possibilita o armazenamento e o acesso a arquivos remotos da mesma forma como se eles estivessem localmente no cliente, possibilitando também que essas informações sejam acessadas de qualquer computador da rede (COULOURIS, 2005).

Um sistema de arquivos distribuído (DFS) é uma implementação de um sistema de arquivos que consiste em locais de armazenamento dispersos fisicamente em uma rede, entretanto, ele provê uma visão de sistema de arquivos tradicional para o usuário (CHOW, 1998). Um DFS possui duas características principais, que devem ser tratadas de forma transparente para os usuários, que é a dispersão e a multiplicidade de usuários e arquivos, ou seja, os múltiplos clientes acessam em diversos locais os arquivos residentes em servidores espalhados pela rede (CHOW, 1998).

A principal característica de um sistema de arquivos distribuído é o fato de ele gerenciar um conjunto de dispositivos de armazenamento dispersos. Seu espaço de armazenamento global é composto por diferentes espaços de armazenamento menores localizado remotamente. Existem configurações de DFS nas quais os servidores são executados em máquinas dedicadas e configurações na qual um computador pode desempenhar o papel de um servidor e de um cliente (SILBERSCHATZ, 2000).

2.1 Conceitos Básicos

No decorrer desta seção, serão apresentados conceitos de sistemas de arquivos distribuídos que servirão de base para a análise das seções subseqüentes.

2.1.1 Sistemas de Arquivos

O sistema de arquivos é uma camada importante do sistema operacional, pois provê uma abstração do armazenamento secundário (discos rígidos, discos magnéticos etc) e é responsável pela nomeação global, o acesso aos arquivos e a organização destes (GALLI, 2000). É ele quem gerencia no sistema operacional o modo com que os arquivos serão estruturados, nomeados, acessados, utilizados, protegidos e implementados (TANENBAUM, 2000).

Um sistema de arquivos fornece serviços de acesso a arquivos para os clientes. Uma interface cliente para este serviço é formada por um conjunto de operações de arquivos primitivas, como criar um arquivo, excluir um arquivo, ler a partir de um arquivo e gravar em um arquivo. O principal componente de hardware que um servidor de arquivos controla é um conjunto de dispositivos de armazenamento secundário local (geralmente discos magnéticos), nos quais os arquivos são armazenados e a partir dos quais eles são recuperados de acordo com os pedidos dos clientes (SILBERSCHATZ, 2000).

2.1.2 Nomeação e Transparência

A nomeação é um mapeamento entre objetos lógicos e físicos. Por exemplo, os usuários trabalham com objetos lógicos de dados representados por nomes de arquivos, enquanto que o sistema manipula blocos físicos de dados, armazenados em trilas de discos, fornecendo ao usuário uma abstração de um arquivo que oculta os detalhes de como e onde ele esta de fato armazenado no disco físico (SILBERSCHATZ, 2000). Dada a localização lógica de um arquivo, ou seja, o caminho até ele, é necessário analisar os componentes deste caminho a fim de encontrar sua localização física. É necessário descobrir em quais blocos de quais discos e de quais servidores se encontra o arquivo em questão (KON, 1994).

Quando os arquivos de um sistema estão distribuídos entre vários servidores localizados em diferentes máquinas, é desejável que a localização destes dados seja transparente aos usuários do sistema (FLOYD, apud KON, 1994). Em um sistema

transparente, uma nova dimensão é adicionada a abstração, a de ocultar o local na rede onde o arquivo esta armazenado:

> “Idealmente, um DFS deve aparecer aos seus clientes como um sistema de arquivos centralizado-convencional. A multiplicidade e a dispersão de seus servidores e dispositivos de armazenamento devem ser transparentes. Ou seja, a interface cliente de um DFS não deve fazer distinção entre arquivos locais e remotos. Cabe ao DFS localizar os arquivos e organizá-lo para o transporte dos dados. Um DFS transparente facilita a mobilidade do usuário trazendo o ambiente do usuário (ou seja, seu diretório inicial) para onde quer que o usuário efetue login” (SILBERSCHATZ, 2000, p. 542) (TRADUÇÃO NOSSA).

Diversas técnicas de transparência, adotadas em sistemas distribuídos e visto na seção 1.1.5, são empregadas parcialmente ou diretamente na implementação de serviços de arquivos (COULOURIS, 2005):

Transparência de Acesso: Esta característica possibilita que as aplicações dos clientes possam acessar arquivos remotos como se estivessem localizados em um sistema de arquivos local, sem haver a necessidade de alterações nos programas.

Transparência de Localização: As aplicações dos clientes devem visualizar um espaço de nomes uniforme, onde os arquivos poderão ser re-alocados, quando necessário, para outros servidores sem alterar seu caminho e nome de acesso.

Transparência de Migração: Permite a movimentação de arquivos entre os servidores do sistema distribuído sem afetar as aplicações e o acesso do cliente.

Transparência de Desempenho: O aplicativo do cliente deve continuar sua execução satisfatoriamente enquanto a carga do serviço variar dentro de uma escala especifica.

Transparência de Escalabilidade: Possibilita a expansão da infra-estrutura do serviço.

2.1.3 Cache

Para aumentar o desempenho no acesso a informação fornecida por um sistema, procura-se armazenar os dados com maior número de solicitação em memória, evitando desta forma uma possível sobrecarga para obter a informação do sistema novamente. Esta técnica auxilia na economia do tempo de processamento, pois para acessar informações remotas, por

exemplo, o sistema estará limitado a velocidade da rede, que, mesmo rápida, estará limitado a velocidade do meio físico do servidor remoto, pois este ainda necessitará procurar e processar os dados solicitados, carregando-os na memória e enviando-os para o cliente (SINGHAL, 1994; CARVALHO, 2003).

Mesmo no acesso a informações locais, a velocidade de acesso à memória é superior a velocidade de acesso ao meio de armazenamento, como um disco rígido, por exemplo, que necessitaria mover o braço do leitor até a trilha em que encontra-se os dados e esperar até que a rotação do disco traga-os à cabeça de leitura (CARVALHO, 2003).

Em sistemas de arquivos distribuídos, o *cache* pode estar presente tanto no cliente, como no servidor, evitando assim que o usuário faça uso desnecessário da rede para obter informações antes já solicitadas, enquanto que o servidor diminui o acesso ao meio físico de armazenamento dos dados para enviá-los ao cliente (CARVALHO, 2003).

2.2 Serviços Oferecidos

Com a finalidade de proporcionar um ambiente de fácil uso para os usuários, escondendo toda a complexidade existente na nomeação global, no acesso aos arquivos dispersos pela rede e na organização destes, os sistemas de arquivos distribuídos fazem uso de três serviços que têm por objetivo controlar e atender estas funcionalidades, que são o serviço de nomes, o serviço de arquivos e o serviço de diretórios, respectivamente (GALLI, 2000).

2.2.1 Serviço de Nomes

O serviço de nomes tem por objetivo indicar a localização de um determinado arquivo no conjunto de computadores do sistema de arquivos distribuído. Caso o nome do arquivo contiver o nome do computador aonde ele esta localizado, como por exemplo “apolo:/tmp/arquivo”, então este serviço não provê transparência de localização. Para prover esta transparência, o nome de um arquivo não deve possuir indícios de sua localização física, pois caso ele mude de lugar ou possua várias cópias, o seu nome ou caminho não precisarão ser alterados (GALLI, 2000).

2.2.2 Serviço de Arquivos

O serviço de arquivos é responsável por fornecer os mesmos serviços e recursos de um sistema de arquivos tradicional, com a diferença que um sistema de arquivos distribuído pode ser acessado de qualquer estação de trabalho de dentro da rede. Esse serviço também cuida das propriedades dos arquivos, como data de criação, data de alteração, tamanho, proprietário, permissões de acesso e outras informações relevantes, além de manter a integridade das operações realizadas nos arquivos (GALLI, 2000).

2.2.3 Serviço de Diretórios

O objetivo do serviço de diretórios é manter a organização dos arquivos armazenados no sistema, fornecendo ao usuário uma interface para que eles possam organizar seus arquivos em uma estrutura hierárquica, através de diretórios e subdiretórios (GALLI, 2000).

2.3 Características Desejadas

Muitos dos conceitos encontrados na implementação de sistemas distribuídos devem ser levados em consideração no desenvolvimento de sistemas de arquivos distribuídos. Inicialmente, os primeiros sistemas de arquivos distribuídos ofereceram recursos de transparência de acesso e localização, emergindo subseqüentemente à preocupação no desenvolvimento de recursos como desempenho, escalabilidade, controle de concorrência, tolerância a falhas e segurança (COULOURIS, 2005).

2.3.1 Atualização Concorrente de Arquivos

Alterações realizadas por um cliente em determinado arquivo não devem interferir com a operação de outro usuário caso ambos estejam manipulando o mesmo documento. A necessidade para gerenciar o acesso simultâneo aos arquivos compartilhados é aceita extensamente e as técnicas de implementação muitas vezes são caras (COULOURIS, 2005).

O maior problema encontrado nas implementações desse tipo de solução é quanto a sincronização dos arquivos, o que inclui leitura e escrita concorrente. A leitura concorrente pode ser implementada facilmente caso não haja escrita concorrente, pois quando um arquivo estiver sendo lido, certamente ninguém poderá alterá-lo. Para implementar a escrita concorrente, deve-se levar em conta que, quando um cliente escreve em um arquivo, todos os

leitores devem ser avisados que o documento foi alterado, tendo-se que tomar cuidado também para que estas alterações não desfaçam as alterações realizadas por outros usuários (COULOURIS, 2005; CARVALHO, 2003).

2.3.2 Replicação de Arquivos

Em um serviço de arquivos com suporte a replicação, o arquivo pode ser representado por várias cópias de seu conteúdo dispersas pelo sistema distribuído, trazendo como beneficio a distribuição da carga de processamento, uma vez que todas as réplicas poderão ser acessadas simultaneamente por clientes diferentes, e realçando a tolerância a falhas, que fará o cliente localizar outro servidor que contenha uma cópia do conteúdo solicitado caso algum nodo venha a falhar (COULOURIS, 2005). A exigência básica de um esquema de replicação é que diferentes réplicas do mesmo arquivo residam em servidores independentes, ou seja, a disponibilidade de uma réplica não é afetada pela disponibilidade das demais (SILBERSCHATZ, 2000). Kon (1994) menciona algumas importantes vantagens que podem existir com a replicação de arquivos:

Caso um disco seja danificado, as informações nele contidas não serão perdidas, podendo ser recuperadas de outros discos em outros servidores;

Se um servidor está momentaneamente inoperante ou inacessível, os seus arquivos podem ser acessados em servidores alternativos, pois haverá uma maior disponibilidade do serviço de arquivos;

Arquivos ou diretórios muito lidos podem ser oferecidos por vários servidores, distribuindo-se a demanda eqüitativamente entre os diversos servidores e aumentando o desempenho global do sistema.

O principal desafio associado a replicação é a sua atualização e manutenção da consistência entre as réplicas dos arquivos nos diversos servidores. Para o usuário, as réplicas de um arquivo denotam a mesma entidade lógica e, dessa forma, uma atualização a qualquer réplica deve refletir nas demais. Os detalhes da replicação devem ser transparentes para os usuários, sendo tarefa do esquema de nomeação mapear um nome de arquivo replicado em um computador específico sem o conhecimento de sua localização pelo cliente (SILBERSCHATZ, 2000; KON, 1994).

2.3.3 Hardware e Sistema Operacional Heterogêneos

Uma das grandes deficiências dos sistemas computacionais tem sido a incompatibilidade tanto de *hardware* quanto de *software* fabricado por diferentes companhias. Cada vez mais, buscam-se padronizações que permitam que máquinas de diferentes fabricantes se comuniquem sem grandes dificuldades (KON, 1994). O sistema de arquivos distribuído deve ser implementado de forma que o servidor e o cliente não necessitem da mesma arquitetura de *hardware* e da mesma solução de *software* para a correta execução do serviço (COULOURIS, 2005).

2.3.4 Tolerância a Falhas

A principal regra de um serviço de arquivos em um ambiente de sistemas distribuídos é garantir o atendimento as requisições dos usuários caso algum servidor venha a falhar (COULOURIS, 2005). Falhas de *hardware* ou de *software* podem ocasionar uma interrupção momentânea no funcionamento de uma máquina. A queda de um nodo, ou uma falha em um canal de comunicação pode levar a uma partição na rede, ou seja, a conversação entre dois pontos da rede é interrompida durante um certo intervalo de tempo (KON, 1994). A grande maioria dos sistemas de arquivos distribuídos é sensível a partições na rede quando estão em operação. Caso o cliente não consiga estabelecer contato com alguns dos servidores, os processos que fizeram solicitações aos serviços de arquivos serão notificados de que as informações solicitadas não estão disponíveis ou simplesmente são bloqueados até que a conexão se estabeleça (KON, 1994).

O método mais utilizado para aumentar a disponibilidade de um serviço de arquivos tem sido a replicação de dados, onde os arquivos são armazenados em dois ou mais servidores e caso um deles não esteja disponível, outro servidor poderá fornecer os serviços solicitados (KON, 1994).

2.3.5 Consistência

Um sistema de arquivos distribuídos deve garantir a consistência dos arquivos de seus usuários. Quando um arquivo é replicado ou recuperado do *cache* de uma estação cliente, por exemplo, podem ocorrer demoras inevitáveis na propagação das modificações, devido a latências de rede, podendo resultar na geração de inconsistências ou até perda das

informações caso o sistema seja frágil a esse tipo de cenário (COULOURIS, 2005; SILBERSCHATZ, 2000).

2.3.6 Segurança

Compartilhar arquivos entre vários ambientes e usuários é uma das vantagens que os sistemas de arquivos distribuídos oferecem. Entretanto, deixar que outros usuários possuam acesso a arquivos confidenciais é um grande problema, sendo necessário adotar mecanismos que garantam que pessoas não autorizadas não tenham acesso a tais informações (COULOURIS, 2005; CARVALHO, 2003).

3 FERRAMENTAS PARA ARMAZENAMENTO DISTRIBUÍDO

No decorrer deste capítulo, serão apresentadas algumas das principais soluções para armazenamento distribuído.

3.1 Network File System

O *Network File System* (NFS), desenvolvido pela Sun *Microsystems* é o sistema de arquivos distribuído mais utilizado pelos usuários de sistemas Unix. Ao disponibilizar a primeira versão do NFS em 1985, a Sun tomou a iniciativa pioneira de divulgar publicamente a especificação de seu protocolo, possibilitando que outras empresas e desenvolvedores pudessem criar clientes e servidores compatíveis. Este protocolo define uma interface de *Remote Procedure Call* (RPC) construída sobre o protocolo *External Data Representation* (XDR). O NFS não é considerado realmente um sistema de arquivos, mas sim uma coleção de protocolos que proporcionam ao cliente o modelo de um sistema de arquivos distribuído. Neste sentido, o NFS é comparável ao *Common Object Request Broker Architecture* (CORBA), que essencialmente, só existe na forma de especificação. Atualmente, é possível encontrar implementações de NFS para quase todas as arquiteturas de computadores e sistemas operacionais, incluindo sistemas não derivados do Unix, como o MVS, o MacOS, o OS/2 e o MS-DOS (KON, 1996; TANENBAUM, 2002).

A idéia principal do NFS é possibilitar que um conjunto de clientes e servidores possam compartilhar um sistema de arquivos em comum, permitindo que um nó seja ao mesmo tempo um cliente acessando arquivos remotos ou um servidor exportando arquivos (TANENBAUM, 2002).

A estrutura do NFS subdivide-se em níveis, conforme apresentado na Figura 3.1. O primeiro nível trata das chamadas do sistema, verificando a sintaxe de cada chamada, validando seus parâmetros e invocando o nível subseqüente, o *Virtual File System* (VFS). O

VFS é um padrão para a interconexão entre diferentes sistemas de arquivos, que suporta diversas implementações de sistemas de arquivos locais, de modo que eles possam ser usados para suportar uma grande variedade de sistemas de arquivos locais e distribuídos (TANENBAUM, 2002).

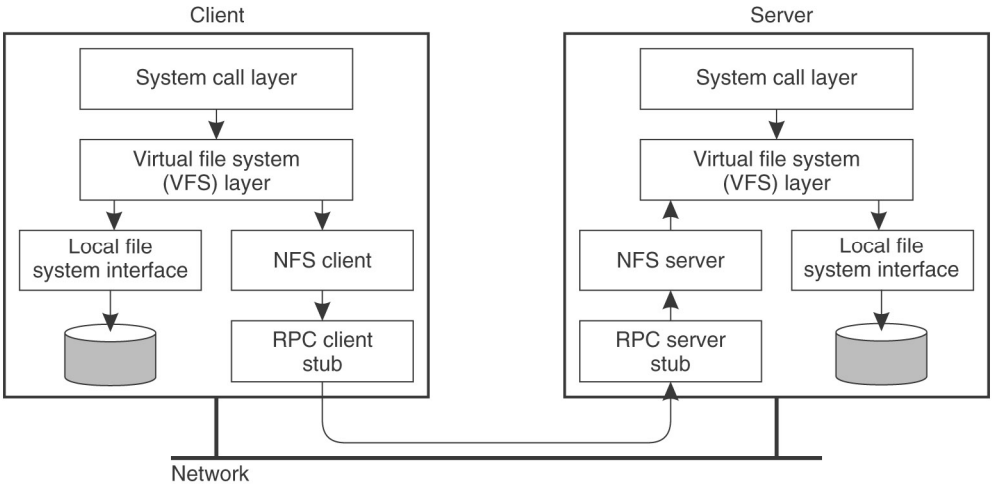

Figura 3.1 - Arquitetura do NFS para sistemas Unix
Fonte: Tanenbaum, 2002, p. 578

O VFS mantém uma tabela para cada sistema de arquivos montado, com uma entrada para cada um dos arquivos abertos, denominada de *v-node*. O *v-node* é utilizado para informar se o arquivo é local ou remoto. Caso o arquivo esteja armazenado localmente, o *v-node* possuirá uma referência para o *i-node* do arquivo local, e caso o arquivo esteja armazenado remotamente, ele conterá o manipulador do arquivo que fornece todas as informações necessárias para acessá-lo (TANENBAUM, 2002).

Quando o cliente efetua uma requisição ao serviço de arquivos, esta solicitação é encaminhada para a interface do VFS, que de acordo com a localização física do arquivo solicitado, remete a requisição para o sistema de arquivos local ou para um cliente NFS. Caso o arquivo esteja armazenado localmente, o NFS faz uso das chamadas de sistema providas pela interface do sistema de arquivos local, e caso o arquivo esteja localizado em outro computador, seu acesso é realizado de forma transparente através do cliente NFS (KON, 1996).

A intenção do protocolo NFS é de ser o mais livre de estado (*stateless*) possível, ou seja, um servidor NFS não precisa manter nenhum tipo de informação sobre o estado da comunicação com seus clientes para oferecer o seu serviço satisfatoriamente. Esta característica traz algumas vantagens imediatas. A característica mais significante é que, quando ocorre uma queda do servidor, ele não perde nenhuma informação sobre o estado da comunicação com os clientes, não precisando, desta forma, gastar tempo na tentativa de reconstituir este estado quando ele recuperar-se da queda. Para o cliente, tudo que ele necessita fazer quando o servidor cair é repetir as chamadas ao serviço até que obtenha resposta (TANENBAUM, 2002). Por outro lado, servidores *stateless* não conseguem gerenciar o acesso concorrente de seus arquivos e desta forma, o NFS não assume a consistência de seu sistema de arquivos, podendo haver casos em que, diferentes clientes possuam diferentes cópias do mesmo arquivo ou diretório no *cache* de seus computadores (KON, 1996).

Quando determinado arquivo for atualizado por um cliente, estas modificações podem não ser percebidas pelos demais usuários durante um período de 6 segundos. Quando ocorre a criação ou exclusão de um arquivo, este fato pode levar 60 segundos para ser percebido pelos demais membros da rede. Caso haja a necessidade de um compartilhamento coerente da informação, algum outro mecanismo - como a passagem de mensagens - deve ser utilizado (KON, 1996; TANENBAUM, 2002).

O NFS suporta transparência de localização e transparência de acesso. Para os clientes fazerem acesso a sistemas de arquivos remotos, eles utilizam uma versão modificada do comando *mount* do sistema Unix, especificando o nome do diretório remoto a ser montado e o nome da máquina do servidor que o armazena. O pedido de montagem é mapeado na RPC correspondente e é encaminhado para o servidor de montagem que executa na máquina servidora específica. A Figura 3.2 ilustra um cliente com dois sistemas de arquivos remotos montados. Os nodos *people* e *users* dos sistemas de arquivos dos computadores *Server 1* e *Server 2* estão montados sobre os nodos *students* e *staff* do computador do cliente. Desta forma, as aplicações que estiverem sendo executadas no computador denominado *Client* podem acessar arquivos residentes em *Server 1* e *Server 2* fazendo uso dos caminhos */usr/students* e */usr/staff* respectivamente (COULOURIS, 2005; SILBERSCHATZ, 2000).

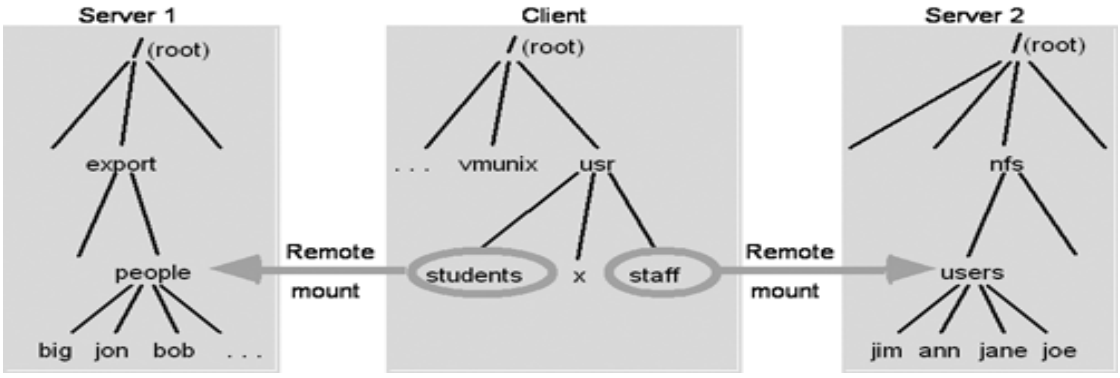

Figura 3.2 - Exemplo da montagem de sistemas de arquivos remotos no NFS
Fonte: Coulouris, 2005, p. 342

3.2 Andrew File System

O projeto Andrew *File System* (AFS) começou na Universidade Carnegie-Mellon em 1983, com o apoio da IBM, e visava o desenvolvimento de um sistema que fosse ideal para o ambiente acadêmico de ensino e pesquisa, oferecendo a cada aluno e professor uma estação de trabalho com um sistema compatível com o Unix BSD, onde os usuários entrariam em qualquer máquina da rede e sua visão do sistema deveria ser a mesma. Desejava-se estender esta transparência de localização a uma rede de 5 a 10 mil estações de trabalho, logo, era essencial tomar o máximo de cuidado com a escalabilidade dos métodos empregados, pois um sistema que se comporta perfeitamente em uma rede experimental de 50 máquinas, pode mostrar-se completamente inadequado para uma rede 100 vezes maior (KON, 1994).

Em termos de arquitetura, o AFS possui alguns pontos em comum com o NFS, como o uso da interface VFS para os clientes possuírem acesso aos arquivos e o uso do RPC como mecanismo de comunicação, o qual utiliza o formato XDR para a representação externa de dados. Outros componentes que fazem parte da arquitetura do AFS são o *Vice* e o *Venus*. *Vice* é o nome dado ao processo servidor que atende as solicitações de serviços de arquivos que chegam. Desde a versão 2 do AFS, o *Vice* é *multi-threaded*, e desta forma, consegue atender a várias solicitações simultâneas eficientemente. O *Venus* é o nome dado ao cliente AFS, o qual é responsável pela interface entre o cliente e o *Vice*. A Figura 3.3 apresenta a distribuição dos processos *Vice* e *Venus* (KON, 1994; COULOURIS, 2005).

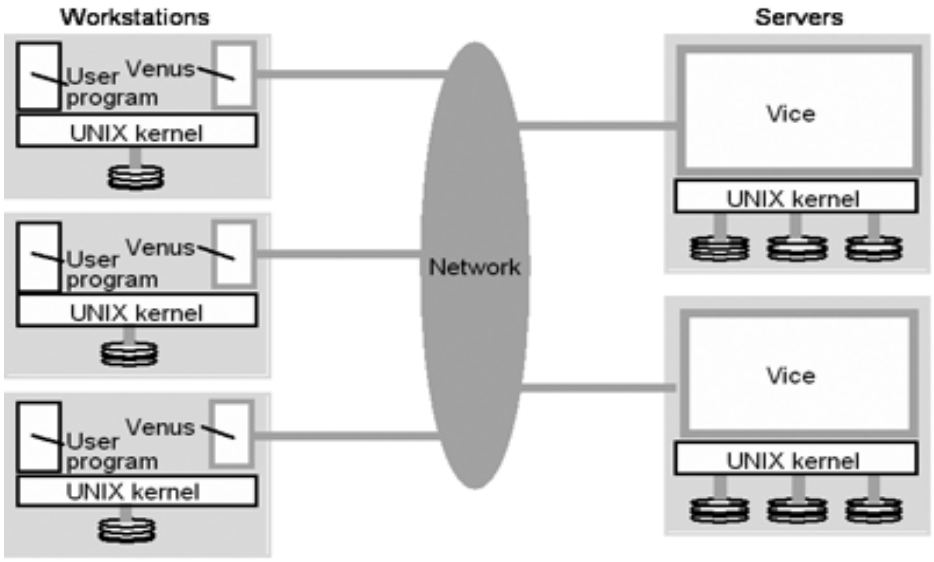

Figura 3.3 - Distribuição dos processos no Andrew *File System*
Fonte: Coulouris, 2005, p. 352

O espaço de nomes do AFS é dividido em duas partes. A primeira é armazenada em discos locais e guardam basicamente informações temporárias (diretório */tmp*), o *cache* do cliente (diretório */cache*) e os arquivos necessários para a inicialização da máquina. A segunda parte (diretório */afs*) é igual para todas as máquinas da rede e contém os arquivos compartilhados mantidos por servidores dedicados (KON, 1994). A Figura 3.4 exemplifica o espaço de nomes de um cliente.

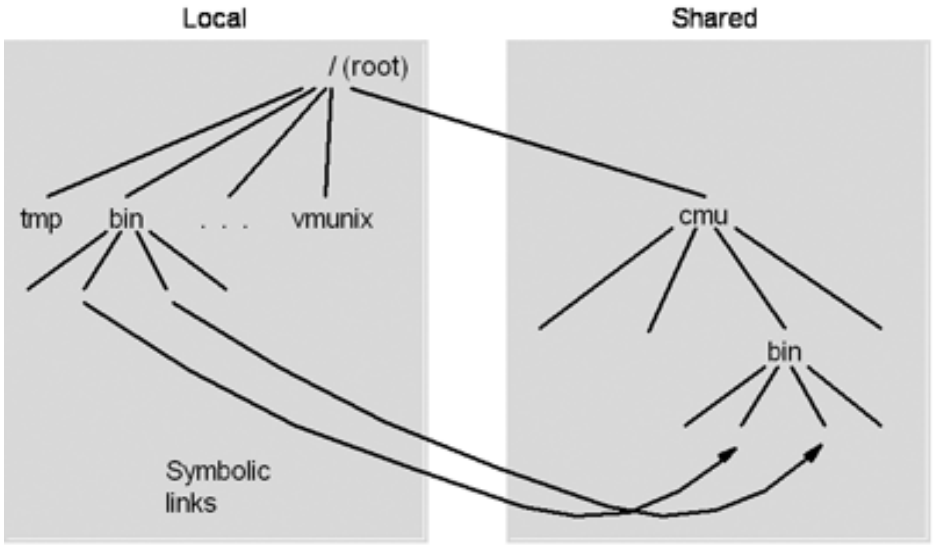

Figura 3.4 - Espaço de nomes vistos de um cliente AFS
Fonte: Coulouris, 2005, p. 352

O AFS disponibiliza recursos para a replicação de volumes e para a realização de cópias de segurança, onde apenas uma das réplicas é considerada alterável, enquanto as demais funcionam apenas para leitura. Um cliente AFS sempre acessa a cópia do volume que está mais próxima. Caso um servidor não responda ao chamado de um cliente, este passa automaticamente a acessar os mesmos dados residentes em outra cópia ativa (KON, 1994).

3.3 Coda File System

O Coda começou a ser desenvolvido em 1987 pela Universidade de Carnegie-Mellon e é descendente da versão 2 do AFS, do qual herdou muitas de suas características arquitetônicas. Seu principal objetivo é fornecer operações desconectadas ao sistema de arquivos para computadores portáteis, que costumam ficar grande parte do tempo fora da rede. Isso provê uma máxima disponibilidade dos arquivos aos usuários, pois os arquivos residirão na máquina local, evitando assim que transtornos como, por exemplo, a queda da rede ou do serviço de armazenamento distribuído, venha a causar contratempo para o usuário. Além da operação desconectada, o Coda utiliza replicação de arquivos como meio de aumentar a disponibilidade (KON, 1994; TANENBAUM, 2002).

O suporte a operações desconectadas surgiu com a proposta de criar um sistema de arquivos resiliente a falhas de rede entre servidores e clientes. O esforço empregado no Coda para lidar com este problema mostrou-se conveniente com o advento de clientes móveis. Quando um usuário atualiza um arquivo, as alterações precisam ser propagadas aos servidores que o armazenam. Caso o cliente esteja conectado ao servidor, esta atualização é feita de forma síncrona, ou seja, a alteração é propagada no momento de sua gravação no cliente. Caso haja problemas nesta comunicação, ao invés de reportar o erro ao usuário, as informações são gravadas localmente em um registro de alterações pendentes e assim que a comunicação com os servidores for restabelecida, estas alterações serão propagadas automaticamente (KISTLER, 1991; BRAAM, 1998).

Os clientes do Coda possuem o sistema de arquivos remoto montado abaixo de */coda*, similar ao seu antecessor. Qualquer arquivo compartilhado por algum servidor Coda, estará disponível neste ponto de montagem para todos os clientes. Diferente do NFS os usuários conectam-se ao serviço Coda e não individualmente nos servidores, que são adicionados como membros do serviço de forma transparente (BRAAM, 1994).

Ao contrário do AFS, no Coda os volumes de dados podem possuir várias cópias que podem ser tanto lidas quanto alteradas pelos clientes. Cada volume possui um grupo de servidores que os replicam, o *Volume Storage Group* (VSG). O subconjunto de servidores do VSG que estão acessíveis por um usuário em um determinado instante é o *Accessible Volume Storage Group* (AVSG) do cliente. Quando o usuário envia para o servidor um arquivo modificado, as alterações são propagadas, em paralelo, para todos os servidores do AVSG e posteriormente, para os demais servidores do VSG (BRAAM, 1994; KON, 1994; TANENBAUM, 2002).

Quando um servidor reintegra a rede, nenhuma providência é tomada para a atualização dos arquivos. Os arquivos somente serão atualizados neste servidor caso seja realizada uma requisição de acesso a uma versão mais recente. Se o cliente descobre que o seu servidor preferencial está com uma versão antiga do arquivo, o cliente solicita a versão mais recente de quem a possuir e emite uma mensagem ao AVGS informando da existência de uma versão desatualizada (KON, 1994; TANENBAUM, 2002).

3.4 Google File System

O Google *File System* (GFS) é um sistema de arquivos distribuído desenvolvido e desdobrado extensamente dentro da empresa Google como plataforma de armazenamento, onde o maior agrupamento de servidores existente para dados provê centenas de *Terabytes* através de milhares de discos rígidos em mais de mil máquinas, sendo acessadas simultaneamente por centenas de clientes. O GFS compartilha de muitos dos mesmos objetivos dos demais sistemas de arquivos distribuídos, tais como escalabilidade, desempenho, confiança e disponibilidade, que foram implementados levando em consideração as necessidades da empresa e o ambiente tecnológico utilizado por ela (GHEMAWAT, 2003).

Em termos de arquitetura, um *cluster* GFS consiste em um único servidor mestre e diversas máquinas *chunkservers* acessadas por inúmeros usuários, conforme ilustrado na Figura 3.5. Cada um destes servidores é um computador rodando o Sistema Operacional Linux com um processo em modo usuário, sendo possível executar um *chunkserver* e um cliente na mesma máquina, desde que haja recursos suficientes e que o baixo nível de confiabilidade seja aceitável (GHEMAWAT, 2003).

Cada arquivo armazenado no sistema de arquivos é dividido em seções de tamanho fixo denominado de *chunks*, onde cada fragmento possui um identificador numérico de 64 *bits* que é definido pelo mestre durante a criação do *chunk*. O mestre é responsável por toda a meta-informação do sistema, tais como as definições de controle de acesso, mapeamento de arquivos para lista de *chunks*, a localização de *chunks*, a recuperação de *chunks* órfãos e a migração de *chunks* entre *chunkservers*, além de realizar manutenções periodicamente a procura de *chunkservers* que possam estar desativados (GHEMAWAT, 2003).

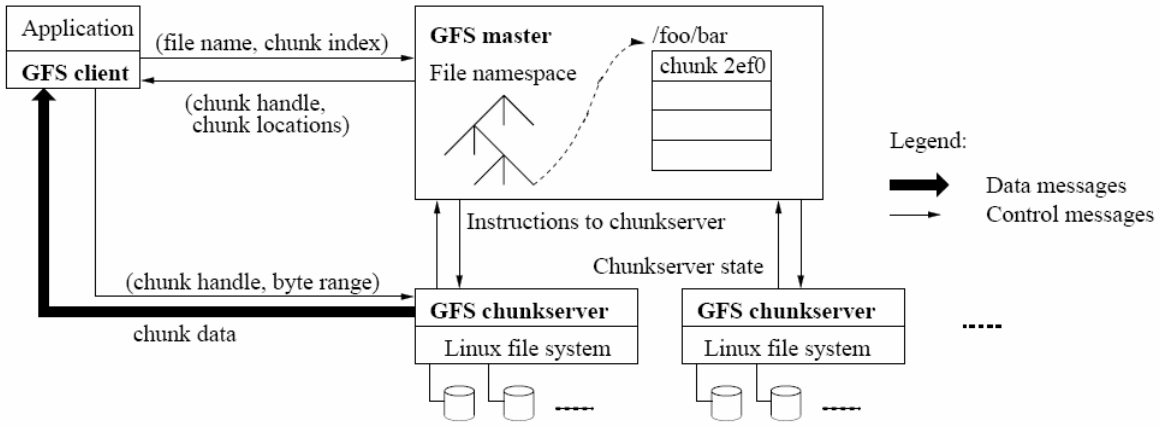

Figura 3.5 - Arquitetura do Google *File System*
Fonte: Ghemawat, 2003, p. 3

Manter um único mestre simplifica a arquitetura e permite a ele decidir sobre a alocação dos *chunks*, bem como sua replicação, usando informações do sistema como um todo. Contundo, o papel do mestre é minimizado nas operações de leitura e escrita, com o intuito de evitar que ele torne-se um gargalo no sistema. Para o processo de leitura de um determinado arquivo, o cliente consulta o servidor mestre para obter a localização dos *chunks* que compõe o arquivo solicitado e de posse desta informação, ele consulta diretamente o servidor *chunkserver* (GHEMAWAT, 2003).

4 EXPERIMENTOS

O objetivo dos experimentos neste trabalho é avaliar as técnicas de replicação, transparência de localização e transparência de acesso utilizado pelas ferramentas de armazenamento distribuído, realizando um estudo aprofundado de como os softwares a serem selecionados implementam estes recursos, mapeando suas vantagens, seus problemas e limitações encontradas durante o processo de simulação.

4.1 Definição do Cenário

Os experimentos e a avaliação das soluções serão realizados em um ambiente composto de no máximo cinco computadores, conforme ilustrado na Figura 4.1, que tem por objetivo aproximar-se de um cenário de utilização real, contemplando a existência de *firewall*, *proxy* e alguns tipos de falhas de comunicação.

No ambiente proposto, os usuários usufruirão de serviços de armazenamento distribuído onde, através de qualquer computador cliente localizado na rede, conseguirão gerenciar seus arquivos de forma transparente, sem precisarem se preocupar com a localização física dos dados e com a utilização de *softwares* específicos para a manipulação destes documentos. Fazendo-se uso de uma analogia, a proposta é similar às redes de energia elétrica, onde toda complexidade referente a geração, transmissão e distribuição é oculta para o usuário, que simplesmente usufrui deste recurso para ligar equipamentos eletrônicos, sem precisar preocupar-se de onde a energia foi gerada e que caminhos ela percorreu para chegar até ele.

Além a transparência de acesso e localização, o ambiente garantirá a replicação dos arquivos dos clientes com o intuito de fortalecer a disponibilidade do serviço.

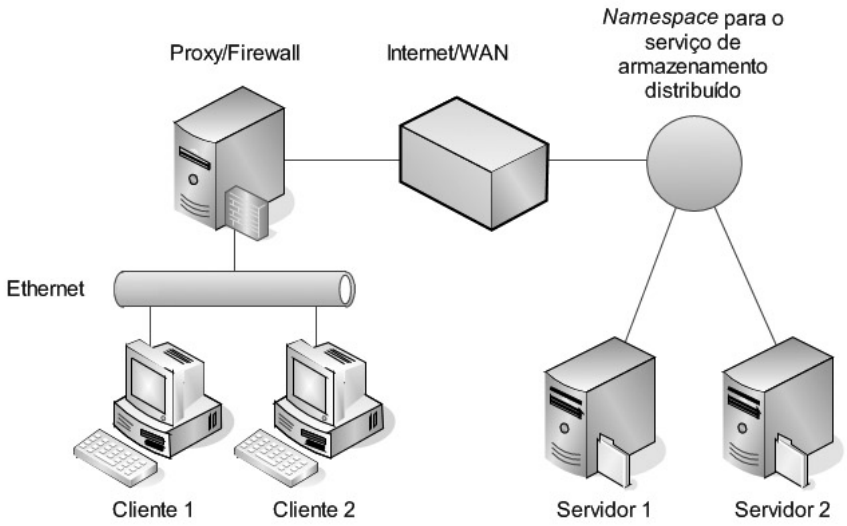

Figura 4.1 - Ambiente de realização dos experimentos
Fonte: Ilustração do autor

Os computadores, denominados na ilustração de Servidor 1 e Servidor 2, serão responsáveis pelo armazenamento dos arquivos. Estas máquinas atenderão as requisições dos clientes através de um *namespace* único e serão agrupadas entre si com uma política de replicação de dados característica da solução implementada. Todas as solicitações dos serviços oferecidos pelo ambiente partirão das estações clientes, representados na ilustração acima pelos computadores Cliente 1 e Cliente 2.

O objetivo do servidor de *proxy* e *firewall* é simular um ambiente com restrições e identificar as portas e protocolos mínimos que necessitam ser disponibilizados aos usuários da rede para fazerem uso dos serviços.

4.2 Definição das Ferramentas

Os critérios para a escolha das soluções que fornecerão o serviço de armazenamento distribuído serão definidos na segunda etapa deste trabalho, levando em consideração o cenário acima descrito, entretanto, será necessário que elas implementem características de transparência de acesso, transparência de localização e replicação de dados, uma vez que estas características são a base do estudo da segunda etapa deste trabalho. As possíveis ferramentas a serem selecionadas são o *Network File System*, o Andrew *File System* e o Coda *File System*, por possuírem as características necessárias para a realização dos experimentos, além de se

tratarem de projetos baseados em Software Livre e possuírem uma vasta citação em referencias bibliográficas.

4.3 Processo de Experimento

A elaboração dos experimentos será compreendida da criação de casos de utilização do ambiente pelo cliente, objetivando a avaliação das técnicas de transparência e replicação empregadas pelas ferramentas. Por exemplo, a avaliação da técnica de replicação de determinada solução, basicamente poderá ser realizada monitorando o comportamento do sistema após a manipulação de determinado arquivo pelo usuário com o intuito de identificar as alterações por ele realizadas em ambos os servidores. Outro teste que pode ser efetuado é simular a falha de um dos computadores que estarão oferecendo o serviço de arquivos e em seguida, realizar uma tentativa de acesso pelo cliente ao arquivo anteriormente manipulado.

A aferição das técnicas de transparência de localização e transparência de acesso podem ser realizadas de forma mais simples em relação a técnica de replicação, entretanto, não deixa de possuir um elevado grau de importância nos sistemas de armazenamento distribuído. Essencialmente, essas técnicas são identificadas pelo cliente quando ocorre seu primeiro contato com o ambiente. Basicamente, a transparência de acesso pode ser identificada caso o usuário não necessite fazer uso de nenhum *software* auxiliar para manipular os arquivos existentes no servidor remoto, enquanto que a transparência de localização pode ser percebida caso o usuário não necessite informar a localização física de um arquivo que deseja acessar, ou seja, o nome do servidor aonde o documento esta armazenado.

4.4 Análise dos Resultados

O objetivo da análise dos resultados é identificar as vantagens, problemas e limitações das ferramentas quanto a utilização das técnicas de replicação, transparência de localização e transparência de acesso, realizando um estudo comparativo entre todas as soluções avaliadas.

CONSIDERAÇÕES FINAIS

O objetivo inicial deste trabalho foi o estudo de conceitos, definições, desafios e problemas gerados com a implementação de sistemas distribuídos e sistemas de armazenamento distribuído, abordando também uma pesquisa por ferramentas que fornecem estes serviços.

Os grandes benefícios oferecidos pelas ferramentas de armazenamento distribuído vêem motivando e fomentando grupos de pesquisas em Universidades e empresas de grande porte, que investem no desenvolvimento de novas técnicas e melhorias para o aperfeiçoamento destas soluções. Diversas ferramentas experimentais foram desenvolvidas nos últimos anos, cada uma delas com uma finalidade específica e com muitas exigências contraditórias. Entretanto, apesar da existência de inúmeras soluções para proporcionar este serviço, diversos assuntos e desafios encontrados na implementação de sistemas distribuídos permanecem problemáticos.

Para ser bem sucedido, o armazenamento distribuído deve tratar um dos grandes desafios de sistemas distribuídos que é a transparência e a replicação. Sendo assim, na segunda parte deste trabalho, serão avaliadas algumas ferramentas de armazenamento distribuído e as técnicas implementadas por elas para resolver os desafios de transparência de acesso, transparência de localização e replicação de dados, identificando suas vantagens, seus problemas e suas limitações. A avaliação será realizada no cenário descrito nesta primeira parte do trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

BEINSYNC. **BeInSync: Secure Remote Access, File Sync and Backup Software.** Disponível em: <http://www.beinsync.com/>. Acesso em: 27 Nov. 2006.

BRAAM, Peter J. The Coda Distributed File System. **Linux Journal**, Seattle, v. 1998, n. 50es, 6 p., Jun. 1998.

CARVALHO, Roberto Pires de**. Sistemas de Arquivos Paralelos e Distribuídos.** São Paulo: 2003. 75 p. Tese (Mestrado em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo, 2003.

CHOW, Randy; JOHNSON, Theodore. **Distributed operating systems and algorithms.** Reading, Massachusetts: Addison Wesley Longman, 1998. 569 p.

CODA. **Coda File System.** Disponível em: <http://www.coda.cs.cmu.edu/>. Acesso em: 27 Nov. 2006.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Distributed systems: concepts and design.** 4. ed. Harlow: Addison Wesley Longman, 2005. 927 p.

DABEK, Frank et al. Wide-area cooperative storage with CFS. In: ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 18., 2001, New York. **Anais...** New York: ACM Press, 2001. p. 202-215.

FOLDERSHARE. **FolderShare: A secure file sharing and collaboration system.** Disponível em: <http://www.foldershare.com/>. Acesso em: 27 Nov. 2006.

GALLI, Doreen L. **Distributed Operating Systems: Concepts and Practice.** New Jersey: Prentice Hall, 2000. 463 p.

GHEMAWAT, Sanjay; GOBIOFF, Howard; LEUNG, Shun-Tak. The Google File System. In: ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 19., 2003, New York. **Anais...** New York: ACM Press, 2003. p. 29-43.

JALOTE, Pankaj. **Fault Tolerance in Distributed Systems.** New Jersey: Prentice Hall, 1994. 432 p.

KIRNER, Claudio; MENDES, Sueli B. T. **Sistemas operacionais distribuídos: aspectos gerais e análise de sua estrutura.** Rio de Janeiro: Campus, 1988. 184 p.

KISTLER, James J.; SATYANARAYANAN M. Disconnected operation in the Coda file system. In: ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 13., 1991, New York. **Anais...** New York: ACM Press, 1991, p. 213-225.

KON, Fabio**. Distributed File Systems Past, Present and Future.** 1996.

KON, Fabio. **Sistemas de Arquivos Distribuídos.** São Paulo: 1994. 154 p. Tese (Mestrado em Matemática Aplicada) - Instituto de Matemática e Estatística, Universidade de São Paulo, 1994.

LODN. **LoDN: Logistical Distribution Network.** Disponível em: <http://loci.cs.utk.edu/lodn/>. Acesso em: 27 Nov. 2006.

MACHADO, Francis Berenger; MAIA, Luiz Paulo. **Arquitetura de sistemas operacionais.** 3. ed. Rio de Janeiro: LTC, 2002. 311 p.

MEDIAMAX. **MediaMax: Free Online Storage.** Disponível em: <http://www.mediamax.com/>. Acesso em: 27 Nov. 2006.

MULLENDER, Sape. **Distributed systems.** 2. ed. New York: ACM Press/Addison-Wesley Publishing Co., 1993. 595 p.

NOTKIN, David et al. Heterogeneous computing environments: report on the ACM SIGOPS workshop on accommodating heterogeneity. **Commun. ACM**, ACM Press, v. 30, n. 2, p. 132-140, Fev. 1987.

OPENAFS. **The Open Andrew File System.** Disponível em: <http://www.openafs.org/>. Acesso em: 27 Nov. 2006.

PAST. **PAST: A large-scale, peer-to-peer archival storage facility.** Disponível em: <http://research.microsoft.com/~antr/past/>. Acesso em: 27 Nov. 2006.

RIBEIRO, Uirá Endy. **Sistemas Distribuídos: Desenvolvendo Aplicações de Alta Performace no Linux.** Rio de Janeiro: Axcel Books do Brasil, 2005. 384 p.

ROUSH, Wade. The Internet Is Your Next Hard Drive**. Technology Review**, 24 Jul. 2006. Disponível em: <http://www.technologyreview.com/read_article.aspx?id=17195&ch=infotech> Acesso em: 27 Nov. 2006.

SIEWIOREK, Daniel P. Architecture of Fault-Tolerant Computers. **Computer**, IEEE Computer Society, v. 17, n. 8, p. 9-18, Ago. 1984.

SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. **Applied Operating System Concepts.** New York: John Wiley & Sons, 2000. 840 p.

SINGHAL, Mukesh; SHIVARATRI, Niranjan G. **Advanced concepts in operating systems: distributed, database, and multiprocessor operating systems.** Nova York: McGraw-Hill, 1994. 522 p.

SINHA, Pradeep K. **Distributed Operating Systems: Concepts and Design.** New York: Wiley-IEEE Press, 1996. 764 p.

SPECTOR, Alfred Zalmon. **Multiprocessing architectures for local computer networks.** Stanford: 1981. 127 p. Tese (Doutorado em Ciência da Computação) - Departamento de Ciência da Computação, Universidade de Stanford, 1981.

STALLINGS, William. **Cryptography and Network Security: Principles and Practice.** New Jersey: Prentice Hall, 2003, 681 p.

TANENBAUM, Andrew S. **Computer Networks.** New Jersey: Prentice Hall, 2003. 384 p.

TANENBAUM, Andrew S.; WOODHULL, Albert S. **Operating Systems Design and Implementation.** 3. ed. New Jersey: Prentice Hall, 2006. 1080 p.

TANENBAUM, Andrew S.; STEEN, Maarten van. **Distributed Systems: Principles and Paradigms.** New Jersey: Prentice Hall, 2002. 803 p.

VERISSIMO, Paulo; RODRIGUES, Luis. **Distributed Systems for System Architects.** Norwell: Kluwer Academic Publishers, 2001. 623 p.