

Sistema de Arquivos Distribuídos



Sistema de Arquivos Distribuídos

- A interface cliente para um sistema de arquivos é composta por um conjunto de primitivas e operações em arquivos (criar, apagar, ler, escrever)
- A interface do cliente deve ser transparente, sem distinguir entre arquivos remotos e locais



Transparência

- Transparência de localização: o nome do arquivo não revela a localização do arquivo
- Independência de localização: o nome do arquivo não precisa ser alterado se a localização do arquivo muda
 - A maioria dos esquemas de nomes não fornece independência de localização
 - A maioria fornece a transparência de localização



Esquema de nomes: absoluto

- Nome da máquina + nome do arquivo
- Vantagens
 - Fácil localização do arquivo
 - Adicionar novos mapeamentos
 - Sem estado global
 - Escalabilidade
- Desvantagens
 - Usuários devem conhecer a localização do arquivo
 - O arquivo é dependente da localização
 - Compartilhamento difícil
 - Sem tolerâncias a falhas



Esquema de nomes: montagem

- Pontos de montagem (NFS – Network File System)
 - Cada máquina tem um conjunto de nomes locais que referenciam posições remotas
 - Mount table (`/etc/fstab`): especifica <remote pathname @ machine name, local pathname>
 - Na inicialização: liga os nomes locais com os arquivos remotos
 - Usuários utilizam o caminho local
 - NFS gerencia o mapeamento



Esquema de nomes: montagem

- **Vantagens:**
 - Localização transparente
 - O nome remoto pode mudar entre reinicializações
- **Desvantagens:**
 - Configuração prévia
 - Um mesmo arquivo pode ter nomes diferentes



Esquema de nomes: esquema global

- Um único espaço de nomes:
 - Exemplos:
 - AFS (CMU's Andrew File System)
 - Sprite (Berkeley)
 - Todos os computadores têm a mesma árvore de diretórios
 - Cliente: obtém a estrutura de diretórios do servidor(s)
 - Quando um arquivo é acessado, o servidor envia para o cliente o arquivo, sendo o mesmo armazenado em cache



Esquema de nomes: esquema global

- **Vantagens:**
 - Nomes – consistência
 - Mesma estrutura em todas as máquinas
 - Fácil movimentação de arquivos (configurado apenas no servidor)
- **Desvantagens:**
 - Consistência de arquivos (cache)
 - Pode limitar a flexibilidade do sistema de arquivos
 - Desempenho



Acesso remoto e cache

- Acesso remoto
 - RPC (remote procedure call): as operações são realizadas no servidor
 - Cache: parte do arquivo é armazenada no cliente
- Questões sobre a cache
 - Quais as partes do arquivo são armazenadas em cache?
 - Quando as alterações são propagadas do cliente para o servidor?
 - O que acontece se dois clientes utilizam um arquivo ao mesmo tempo?



Cache de arquivos remotos

- **Disco:**
 - + Reduz o tempo de acesso (comparado com o acesso remoto)
 - + Seguro se o nó falha
 - Consistência entre a cópia remota e o servidor
 - O cliente precisa de um disco
- **Memória principal:**
 - + Acesso rápido
 - + Funciona em clientes sem disco
 - Consistência entre a cópia remota e o servidor
 - O tamanho da cache deve ser menor
 - Sem tolerância a falhas

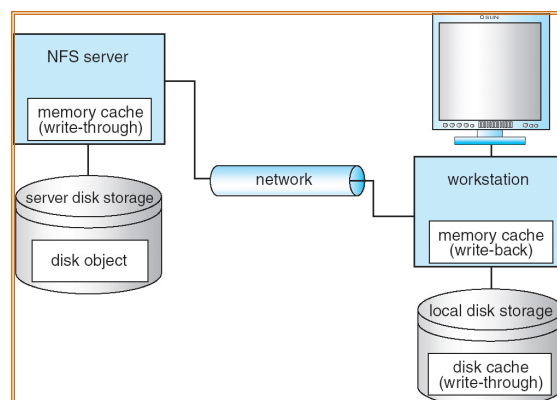


Políticas para consistência da cache

- Escrita direta (write through)
 - Confiável
 - Baixo desempenho
- Escrita adiada (delayed write)
 - Escreve somente na cache
 - As alterações enviadas periodicamente
 - Rápido
 - Reduz o acesso à rede
 - Se a máquina do cliente é reinicializada os dados são perdidos
 - Variação: write-on-close



Consistência da cache



Consistência da cache

- Iniciada pelo cliente
 - Cliente contacta o servidor e verifica a consistência
 - Verificar
 - cada acesso
 - intervalos
 - quando o arquivo é aberto
- Iniciada pelo servidor
 - O cliente avisa o servidor que um determinado arquivo foi aberto (modo escrita, leitura)
 - o servidor detecta conflitos potenciais e invalida a cache
 - Servidor armazena quais são os arquivos que estão na cache dos clientes (statefull)



Servidor com estado (stateful)

- Mecanismo
 - Cliente solicita a abertura de um arquivo
 - O servidor busca as informações sobre o arquivo no disco, armazena na memória, e retorna para o cliente o identificador único
 - O identificador é utilizado para os acessos subsequentes
 - O servidor pode retirar da memória principal as informações do arquivo depois de um tempo de inatividade do cliente
- Aumento de desempenho
 - Poucos acessos no disco
 - Um servidor com estado pode reconhecer um acesso sequencial e fazer leituras



Servidor sem estado (stateless)

- O servidor não armazena informações sobre os arquivos abertos pelos clientes
- Cada solicitação é auto-contida
 - Arquivo e posição
- Sem necessidade de estabelecer ou fechar as conexões



Stateful vs Stateless Service

- Recuperação de falhas
 - Um servidor com estado perde todas as informações voláteis após uma parada
 - Recuperar o estados através de um protocolo de recuperação com os clientes, ou abortar as operações que estavam sendo realizadas quando a parada ocorreu
 - Servidor precisa estar ciente de falhas no cliente, para liberar os recursos alocados pelo cliente
 - Falhas no servidor não são notadas
 - O servidor reinicializado pode responder as requisições normalmente



Diferenças

- Penalidades na implementação de um servidor sem estado robusto:
 - Mensagem de requisição longas
 - Tempo de processamento
- Alguns ambientes requerem servidores com estado
 - Servidor com consistência de cache iniciada pelo servidor, pois ele armazena os arquivos que estão no cliente
 - UNIX utiliza descritores de arquivo e conseqüentemente o deslocamento, servidores devem manter uma tabela com os arquivos abertos, o *inode* e os deslocamentos



AFS – Andrew File System

- O sistema de arquivos faz parte de um ambiente de computação distribuída (Andrew) desenvolvido na Universidade Carnegie-Mellon University (desde 1983), foi adquirido pela IBM e lançado como **Transarc DFS**,
- Disponível como código aberto: OpenAFS
- AFS tenta resolver questões como espaço de nomes uniforme, compartilhamento de arquivo independente de localização, cache no cliente (com consistência), e autenticação segura
 - Cache no servidor (utilizando replicas), alta disponibilidade
 - Suporta até 5000 clientes



ANDREW (2)

- Os clientes tem um dois espaços de endereçamento: **local name space** e **shared name space**
- Servidores dedicados denominados *Vice*, exportam o espaço de nomes compartilhado para os clientes como o homogêneo, idêntico e com transparência de localização da hierarquia de arquivo
- O espaço local de nomes é a raiz do sistema de arquivos da estação de trabalho, na qual o espaço de nomes compartilhado é montado
- Estações de trabalho executam o protocolo *Virtue* para comunicar com o *Vice*, e requerem um disco local para armazenar o espaço de nomes
- Os servidores são responsáveis (de forma coletiva) pelo armazenamento e gerenciamento do espaço de nomes compartilhado



ANDREW (3)

- Clientes e servidores são estruturados em clusters interconectados por uma rede LAN
- O cluster consiste em uma coleção de estações de trabalho e um servidor do cluster
- O mecanismo chave no sistema é a cache do arquivo completo no cliente
 - Quando um arquivo é aberto, o mesmo é armazenado na cache, no disco local do cliente



ANDREW Shared Name Space

- Volumes no Andrew são as menores unidades associadas aos arquivos de um cliente
- Um **FID** identifica um arquivo *Vice* ou diretório – Um fid tem 96 bit, sendo composto por 3 campos:
 - Número de volume
 - **vnode number** – índice na tabela de inodes
 - **uniquifier** – permite o reuse de número de vnode, desta forma deixando certas estruturas compactos
- FIDs são transparentes em termos de localização, portanto, movimentações de arquivos de servidor para servidor, não invalida o conteúdo dos diretórios armazenados na cache
- A informação sobre a localização é armazenada em termos de volume, sendo replicada em cada servidor



ANDREW- Operação em arquivos

- Andrew coloca arquivos inteiros na cache
 - Um cliente interage com o servidor *Vice* apenas nas operações de abrir e fechar arquivo
- *Venus* (cliente) – armazena arquivos na cache quando são abertos, e envia as cópias modificadas dos arquivos de volta quando eles são fechados
- Leitura e escrita de arquivos é realizada pelo SO do cliente, sem intervenção do *Venus*
- A cache no *Venus* armazena também o conteúdo dos diretórios e links simbólicos
- Exceção: modificações em diretórios são realizadas diretamente no servidor responsável pelo diretório



ANDREW- Implementação

- Os clientes são interfaceados diretamente com o kernel UNIX usando chamadas de sistema
- Venus realiza a tradução de nomes parte por parte
- O sistema de arquivos UNIX é usado como sistema de armazenamento de baixo nível, tanto nos servidores como em clientes
 - Cache no cliente: é um diretório na estação local
- Cliente e Servidor acessam os arquivos diretamente usando os inodes, evitando o custo de traduzir o caminho/nome do arquivo para o inode



ANDREW- Implementação (2)

- Venus gerencia duas caches separadas:
 - status
 - data
- Algoritmo LRU: usado para deixar a cache com um tamanho limitado
- A cache de status é armazenada na memória virtual, permitindo uma resposta rápida às requisições *stat*
- A cache de dados é armazenada no disco local, porém o UNIX tem também mecanismos de buffers de blocos de disco na memória, que são transparentes ao Venus

