Sumário

- Distribuição.
- Arquitectura.
- Gestão da memória escondida (“cache”).
- Exemplos.

Serviço do sistema de ficheiros

- Nomes dos ficheiros.

  Geralmente, estrutura de nomes hierárquica: ficheiros e directorias.

- Operações sobre os ficheiros:

  Abrir, fechar, escrever, ler, posicionar, etc.

Distribuição nos sistemas de ficheiros

- Aceder a um ficheiro a partir de uma máquina remota.

  E não só na máquina que guarda o ficheiro no seu disco local.

- Nem todos os ficheiros podem estar na mesma máquina.

  Diferentes “partes” do sistema de ficheiros estão em máquinas diferentes.

- Alguns ficheiros podem estar replicados.

Arquitectura distribuída

- Servidor.

  Processo ou conjunto de processos que têm acesso directo ao disco.

  A cada servidor está associado um conjunto de ficheiros.

- Cliente(s).

  Processos que utilizam os serviços através do(s) servidor(es).

  Necessitam de descrobrir qual ou quais os servidores que possuem cópias do ficheiro.

Modelos de acesso

- Acesso remoto.

  Todas as operações são realizadas remotamente, através de RPC para os servidores correspondentes.

- Cópia local.

  O cliente obtém uma cópia local do ficheiro que deseja ler ou escrever.

  Depois de alterado, o ficheiro é copiado de novo para o servidor.

Estado nos servidores

- Servidores com estado.

  Guardam informação acerca de quais os ficheiros abertos.

  Mais eficientes, mas mais díficeis de replicar e de recuperar em caso de falha.

- Servidores sem estado.

  Cada pedido não depende dos anteriores.

  Menos eficientes.

  Fácil tolerar falhas.

Modelos de coerência

- Mais uma vez, é necessário definir qual é o comporatmento correcto do sistema de ficheiros.
- Considerar cada operação atómica: *coerência sequêncial.*
- Agrupar sequências de operações: *serializabilidade.*

  Ex: desde o momento em que se abre o ficheiro até voltar a fechar (semântica tipo sessão).

- Modelos mais fracos: *para aumentar a disponibilidade.*

Padrões de acesso

- Foram estudados padrões de acesso em ambientes universitários [Satyanarayanan81].
- Conclusões:

  Maioria dos ficheiros são pequenos.

  Leitura muito mais frequente.

  Leituras sequenciais maioritárias.

  Muitos ficheiros temporários.

- Serão estes resultados válidos noutros contextos/épocas?

Memória escondida ("cache")

- Manter automaticamente cópias do ficheiro que está acedido em locais mais "próximos".
- Memória volátil do servidor.
- Memória volátil do cliente.
- Disco local do cliente (se existir).

Como manter a coerência com memória escondida

- Memória escondida só no servidor.

  Todos os acessos têm de passar na rede.
- Actualizar a cópia local e a cópia central simultaneamente.

  Gestão da replicação do tipo "escreve-todos-lê-um".

  Eficiente na leitura mas com penalização na escrita.
- Actualizar só quando o ficheiro é escrito.

  Funciona bem se se considerar uma "sessão" atómica.

Optimização

- Dado que muitos ficheiros são temporários.

  São escritos e apagados logo de seguida.
- Pode-se atrasar a cópia para o servidor.

  O ficheiro pode ser apagado (e já não é necessário realizar a cópia).

  Podem-se copiar vários ficheiros de uma vez (eventualmente, na mesma mensagem).
- Problema:

  Se as aplicações comunicam por outros meios, o modelo de coerência do sistema de ficheiros pode não ser congruente.

Replicação

- Os ficheiros são um caso particular da replicação de dados.

  Qualquer das aplicações discutidas anteriormente é válida.
- Por vezes pretende-se replicar um sistema de ficheiros já e-xistente, alterando o mínimo possível o cliente.

  O modelo primário-secundário é apropriado, porque cada cliente só fala com um primário em cada instante.

Partições e mobilidade

- Em sistemas de grande escala as partições são frequentes.

  Manter disponibilidade dos ficheiros.

- Cada vez existem mais computadores portáveis.

  A portabilidade pode ser considerada como uma partição.

- As operações em directorias têm uma semântica bem definida.

  É mais fácil suportar a actividade em diferentes partições e reconciliar automaticamente as réplicas.

NFS: arquitectura

- Network File System.

  Originalmente desenvolvido pela Sun e hoje disponível para um elevado número de arquitecturas e sistemas operativos.

- Modelo cliente servidor.

  Cada servidor exporta um conjunto de directorias.

  Os clientes montam as directorias remotamente exportadas em "directorias" locais: qualquer acesso é automáticamente transformado num acesso remoto.

NFS: protocolos

- As mensagens trocadas entre os clientes e os servidores estão bem definidas.

  Conjunto de protocolos conhecidos.

  Especifica qual o formato dos pedidos, respostas, e quais as respostas possíveis para cada pedido.

- Clientes e servidores só necessitam de respeitar o protocolo.

NFS: "mouting"

- Explicíto.

  Uma directoria remota é montada localmente comunicando com o servidor, o qual devolve uma chave necessária para interacções futuras.

- A pedido ("automounting").

  O protocolo só é executado quando se tenta aceder a um ficheiro nessa directoria.

  É possível indicar uma lista de servidores alternativos (que devem ter uma réplica perfeita da árvore).

  Usado principalmente para árvores de leitura (sistema operativo, etc).

NFS: acesso

- Acesso concretizado por chamadas a procedimentos remotos.

  Todas as chamadas do unix são suportadas, excepto o "open" e "close" (estas últimas, têm uma semântica diferente).

  O servidor não necessita de manter estado acerca de quais os ficheiros abertos, nem em que posição o cliente se encontra.

  Esta informação é mantida pelo cliente.

- Protocolo original não seguro.

  Pressupunha-se que a informação enviada pelo cliente era correcta (nome e grupo do utilizador). Hoje é possível usar criptografia para autenticar os clientes.

NFS: concretização

- Lê sempre grandes blocos de dados (tipicamente 8K).

  Leitura antecipada: permite têr a informação localmente, quando esta é necessária.

- Os ficheiros são guardados localmente.

  Periodicamente (30s) as alterações são enviadas para o servidor.

  Cópias para leitura são descartadas periodicamente.

  Sempre que se abre um ficheiro, é contactado o servidor para verificar se a cópia é recente.

Sprite: nomes e localização

- Sistema de ficheiros experimental desenvolvido em Berkeley (1988).
- Espaço único de nomes.

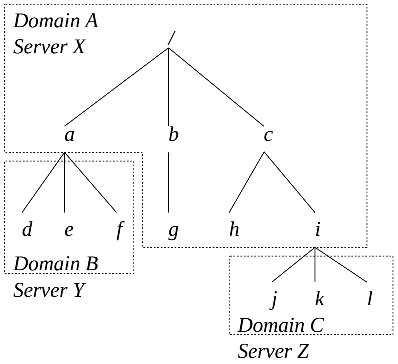
  O mesmo ficheiro tem o mesmo nome em qualquer lado.

- Cada servidor gere uma sub-árvore de ficheiros.

  Identificada pelo prefixo da directoria raiz dessa sub-árvore em relação à raiz global.

- É mantida a associação entre prefixos e servidores.

Sprite: ilustração



/	X
/a	Y
/c/i	Z

Sprite: gestão de prefixos

- Cada cliente mantém uma tabela de prefixos local.

  Inicialmente esta tabela encontra-se vazia.

- Quando se acede a um ficheiro procura-se na tabela o prefixo mais longo que emparelhe com o nome do ficheiro.

  Se não existir, é feita uma difusão com o nome do ficheiro para encontrar o servidor correspondente.

  O servidor que tem o ficheiro, responde com o seu prefixo, endereço, etc.

Sprite: escrita diferida

- Cada cliente mantém uma cópia "escondida" dos ficheiros que acede.
- Ficheiros alterados nos clientes não são propagados de imediato para o servidor.

  Ficheiros alterados recentemente não são copiados, na expectativa de serem alterados de novo.

  Ficheiros não alterados à mais de 30s são copiados para o servidor.

  Caso necessite de fornecer informação actualizada a outro cliente, o servidor também pode pedir a um cliente para copiar o seu estado.

Sprite: escrita concorrente

- Se o servidor detecta escritas concorrentes ao ficheiro, informa os clientes que não devem manter cópias escondidas desse ficheiro.

  Toda a sincronização é feita no servidor.

AFS

- Andrew File System, desenvolvido em CMU.

  Projecto começou em 1983.

- Destingue clientes e servidores.

  Os servidores gerem um espaço único de nomes.

  Os clientes guardam cópias "escondidas" dos ficheiros que estão a aceder.

- Diversas versões do sistema.

  AFS-1 (primeira versão) até AF-3 comercializado pela Transarc.

AFS: gestão da memória escondida

- Primeira versão: pessimista.

  Validade da cópia local verificada a cada acesso.

- Segunda versão: "callback".

  O servidor é responsável por invalidar a cópia local do cliente quando o ficheiro é alterado.

Coda

- Evolução do Andrew FS, desenvolvido em CMU.
- Desenhado para manter elevada disponibilidade e suportar clientes móveis.
- Escalabilidade.

  Para não sobrecarregar os servidores, muito funcionalidade executada nos clientes.

- É permitido o progresso em todas as partições.

Coda: arquitectura

- Ficheiros agrupados em volumes.
- Cada volume pode estar replicado em diversos servidores.
- Cada cliente mantém, para cada volume, uma lista de servidores que consegue contactar no momento.

  Clientes diferentes podem aceder a servidores diferentes.

Coda: acesso

- O cliente tenta descobrir qual o servidor acessível que tem uma versão mais recente do ficheiro.

  Se vários estiverem nesta situação, escolhe o servidor mais próximo.

- Se houver versões diferentes.

  O cliente avisa os servidores desactualizados, que devem obter novas cópias.

- O cliente mantém uma cópia local desse ficheiro.

  O servidor é responsável por avisar o cliente caso alguém altere o ficheiro (callback).

Coda: coerência

- O cliente actualiza o conjunto de servidores acessíveis.

  A actualização é utilizada para verificar a coerência do servidores.

- Se descobre que um servidor que ficou acessível tem informação mais recente, invalida a sua memória escondida.

  Cancela os callback pendentes.

- É possível que o servidor responsável pelo callback não esteja acessível de um outro cliente que altere os dados.

  O cliente têm de verificar a coerência da memória escondida periodicamente.

Coda: operação desligada

- Se um cliente não consegue aceder a nenhuma das réplicas do volume, diz-se desligado.

  As operações são permitidas na cópia escondida local ao cliente.

- O utilizador pode especificar quais os ficheiros que devem ser mantidos em memória local ("hoarding").

  Para os poder utilizar enquanto está desligado.

  O sistema decide quais os ficheiros de que deve manter cópias, com base na informação explicita e com base na história de utilização.

Coda: operação desligada

- Durante operação desligada o sistema mantém um histórico de todas as operações realizadas.

  De modo a mais tarde poder fazer a integração com os servidores.

- O sistema tenta reconciliar automáticamente as directorias.

  Não se consegue reconciliar automaticamente os ficheiros.

  Quando não consegue, cria uma directoria auxiliar e requer correcção manual.

Resumo

- Modelo cliente servidor.
- Espaços de nomes.
- Modelos de coerência.
- Exemplos.

  NFS, SPRITE, AFS, Coda.