GBC074 – Sistemas Distribuídos

Sistemas de arquivos distribuídos

Sistemas de arquivos distribuídos

- Requisitos - Transparência:
- Transparência de **acesso**
  - Programas clientes não devem conhecer a distribuição dos arquivos
  - Programas operam em arquivos locais e remotos

- Transparência de **localização**
  - Clientes devem usar um espaço de nome de arquivo uniforme
  - Arquivos podem ser realocados sem alterar seus nomes de caminho

Sistemas de arquivos distribuídos

- Requisitos - Transparência:
- Transparência de **desempenho**
  - Distribuição não deve prejudicar o desempenho (comparado a sistemas de arquivos centralizados)

- Transparência de **escala**
  - Serviço pode ser expandido por crescimento incremental

Sistemas de arquivos distribuídos

- Requisitos - Outros:

- Atualizações simultâneas de arquivos
  - Como lidar com vários clientes acessando o mesmo arquivo?
  - Técnicas conhecidas em sistemas de arquivos centralizados, mas difíceis ou caras de implementar em ambientes distribuídos
  - Tendência recorrente: siga os padrões modernos do UNIX para fornecer bloqueio de nível de arquivo ou registro obrigatório ou consultivo

Sistemas de arquivos distribuídos

- Requisitos - Outros:

- Replicação de arquivos
  - A replicação permite aumentar a escalabilidade e a disponibilidade
  - Mas difícil de implementar
  - Poucos sistemas de arquivos suportam replicação, mas a maioria suporta cache
- Heterogeneidade de hardware e sistema operacional
  - A interface de serviço deve permitir a implementação de software cliente e servidor para diferentes sistemas operacionais e computadores

Sistemas de arquivos distribuídos

- Requisitos - Outros:

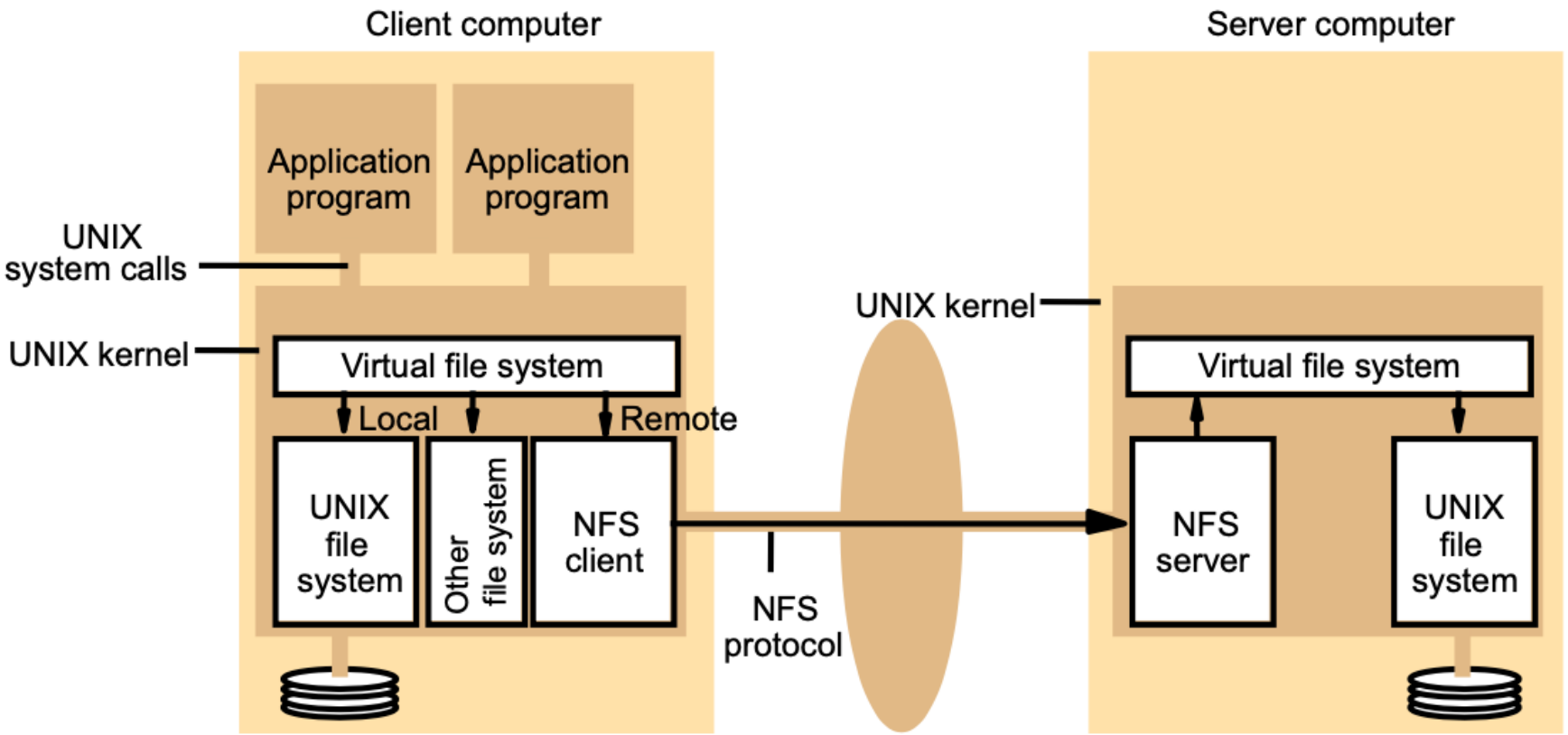
- Consistência
  - Idealmente, imitar a consistência de sistemas centralizados (1 cópia)
  - Difícil devido à distribuição, replicação, armazenamento em cache, ...
- Segurança
  - Autenticação além dos mecanismos de controle de acesso
  - Criptografia de dados

Sistemas de arquivos distribuídos - NFS

- Sistema de Arquivos de Rede (NFS) da SUN
- Introduzido em meados dos anos 80
- Altamente bem sucedido, tanto técnica como comercialmente
- Interfaces-chave colocadas em domínio aberto e amplamente implementadas (Windows, Mac OS, Linux)
- Acesso transparente a arquivos remotos
- Qualquer computador pode ser um servidor, exportando seus arquivos

Sistemas de arquivos distribuídos - NFS

- Arquitetura



Sistemas de arquivos distribuídos - NFS

- Servidor NFS
  - Reside no kernel em cada computador servidor NFS
  - Os módulos cliente e servidor se comunicam usando RPC (RPC desenvolvido para ser usado com NFS)
- Sistema de arquivos virtuais
  - NFS fornece transparência de acesso
  - Distingue entre arquivos locais e remotos
  - Identificadores de arquivos do NFS: manipuladores de arquivos (*file handlers*)
    - ID do sistema de arquivos + número do arquivo i-node + núm. de geração do i-node
    - Substitui os i-nodes do Unix por v-nodes

Sistemas de arquivos distribuídos - NFS

- Integração do cliente
  - Emula a semântica do sistema de arquivos Unix
  - Integrado no kernel (não disponível como biblioteca)
    - Os programas do usuário podem acessar arquivos via Unix sem recompilação e recarregamento
    - Módulo único atende a todos os processos do usuário (+cache compartilhado)
- Autenticação de controle de acesso
  - O servidor NFS é sem estado
  - O servidor deve verificar o id do usuário em cada solicitação

Sistemas de arquivos distribuídos - NFS

- Interface do servidor

lookup(dirfh, name) -> fh, attr	Returns file handle and attributes for the file *name* in the directory *dirfh*.
create(dirfh, name, attr) -> newfh, attr	Creates a new file name in directory *dirfh* with attributes *attr* and returns the new file handle and attributes.
remove(dirfh, name) status	Removes file name from directory *dirfh*.
getattr(fh) -> attr	Returns file attributes of file *fh*. (Similar to the UNIX *stat* system call.)
setattr(fh, attr) -> attr	Sets the attributes (mode, user id, group id, size, access time and modify time of a file). Setting the size to 0 truncates the file.
read(fh, offset, count) -> attr, data	Returns up to *count* bytes of data from a file starting at *offset*. Also returns the latest attributes of the file.
write(fh, offset, count, data) -> attr	Writes *count* bytes of data to a file starting at *offset*. Returns the attributes of the file after the write has taken place.
rename(dirfh, name, todirfh, toname) -> status	Changes the name of file *name* in directory *dirfh* to *toname* in directory to *todirfh*
link(newdirfh, newname, dirfh, name) -> status	Creates an entry *newname* in the directory *newdirfh* which refers to file *name* in the directory *dirfh*.

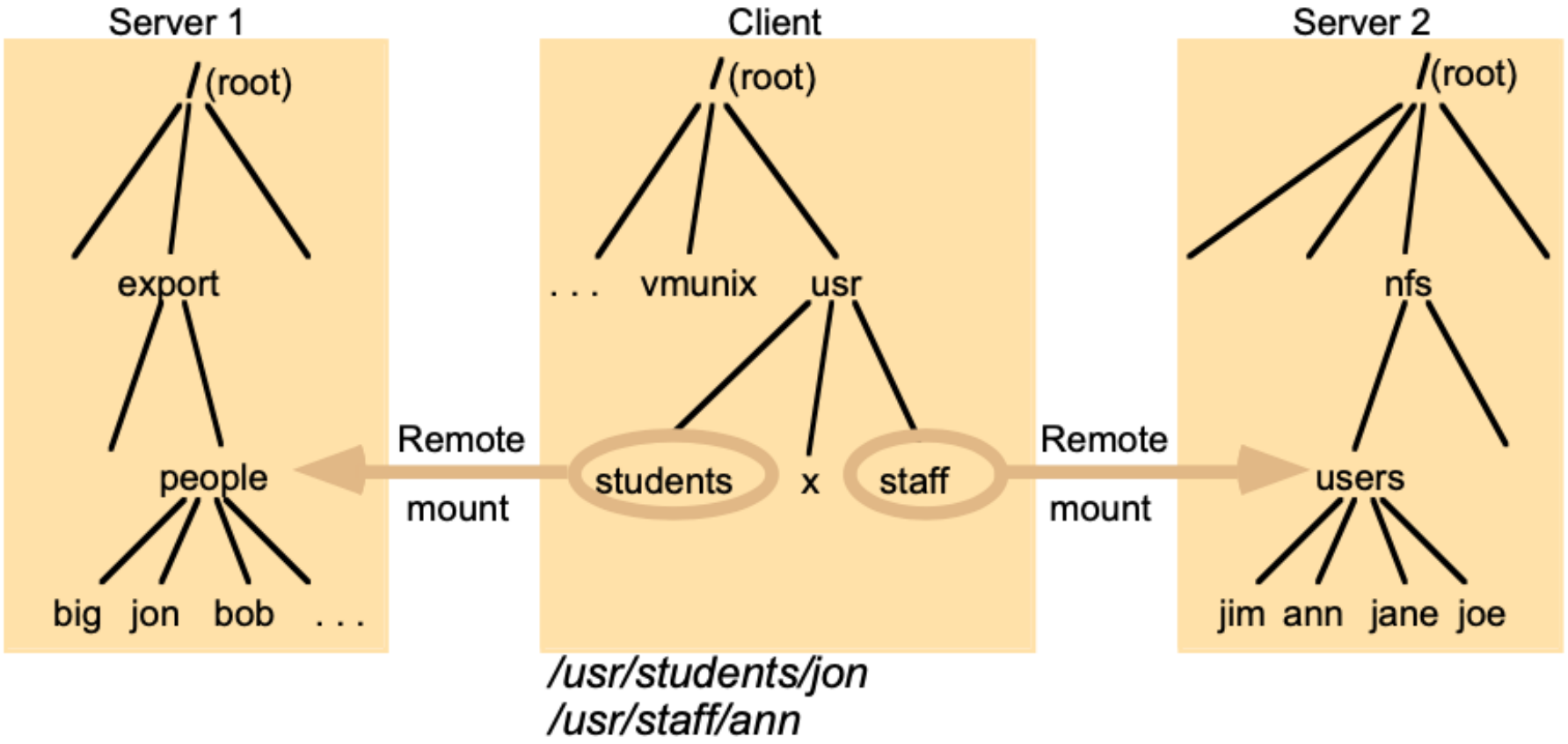
Sistemas de arquivos distribuídos - NFS

- Interface do servidor (continuação)

symlink(newdirfh, newname, string) -> status	Creates an entry *newname* in the directory *newdirfh* of type symbolic link with the value *string*. The server does not interpret the *string* but makes a symbolic link file to hold it.
readlink(fh) -> string	Returns the string that is associated with the symbolic link file identified by *fh*.
mkdir(dirfh, name, attr) -> newfh, attr	Creates a new directory *name* with attributes *attr* and returns the new file handle and attributes.
rmdir(dirfh, name) -> status	Removes the empty directory *name* from the parent directory *dirfh*. Fails if the directory is not empty.
readdir(dirfh, cookie, count) -> entries	Returns up to *count* bytes of directory entries from the directory *dirfh*. Each entry contains a file name, a file handle, and an opaque pointer to the next directory entry, called a *cookie*. The *cookie* is used in subsequent *readdir* calls to start reading from the following entry. If the value of *cookie* is 0, reads from the first entry in the directory.
statfs(fh) -> fsstats	Returns file system information (such as block size, number of free blocks and so on) for the file system containing a file *fh*.

Sistemas de arquivos distribuídos - NFS

- Serviço de montagem (*mount service*)



Sistemas de arquivos distribuídos - NFS

- Cache do **servidor**:
  - Indispensável para um desempenho adequado

- No Unix convencional
  - **Leitura antecipada (*read-ahead*)** antecipa os acessos de leitura e busca as páginas que seguem as lidas recentemente
  - **Gravações com atraso (*delayed-write*)** no disco somente quando a página de buffer é necessária para outra página
  - Operação de **sincronização (*sync*)** libera as páginas alteradas para o disco a cada 30 segundos

Sistemas de arquivos distribuídos - NFS

- Cache do **servidor**:
  - Indispensável para um desempenho adequado

- No NFS
  - Leituras processadas usando leitura antecipada
  - Opção 1:
    - Write-through grava dados antes da resposta enviada ao cliente
  - Opção 2:
    - Gravar apenas na memória; *commit* garante dados no disco
  - ***Commit*** é uma operação adicional (NFS versão 3) para superar o gargalo de desempenho causado por write-through em servidores que recebem um grande número de gravações

Sistemas de arquivos distribuídos - NFS

- Cache do **cliente**
  - Clientes armazenam em cache os resultados de leitura, gravação, *getattr*, *lookup* e operações readdir
  - Risco de diferentes versões em diferentes caches de clientes:
    - Gravações do cliente não atualizam imediatamente todos os caches
  - Clientes fazem *poll* no servidor para atualizar dados em cache
  - Intervalo de sondagem definido de forma adaptável para arquivos e diretórios
    - Arquivos normalmente no intervalo de 3 a 30 segundos
    - Diretórios na faixa de 30 a 60 segundos

Sistemas de arquivos distribuídos - NFS

- Performance
  - As primeiras medições (ou seja, 1987) não mostraram perda de desempenho quando comparado a arquivos armazenados em discos locais
  - Dois problemas
    - Uso frequente da chamada getattr para buscar timestamps
    - Baixo desempenho de gravação (por causa do write-through)
  - Gravações são raras no Unix (5% de todas as chamadas) e o mecanismo de confirmação (commit) reduz o problema com as gravações
  - Atualmente (2012), os resultados mostram taxas de transferência de 12.000 operações por segundo em nós de CPU únicos

Sistemas de arquivos distribuídos - NFS

- Resumo:

- **Transparência de acesso**: não são necessárias modificações nos programas existentes
- **Transparência de localização**: obtida pela montagem de sistemas de arquivos remotos no espaço de nomes local
- **Escalabilidade**: NFS pode lidar com grandes cargas, mas tem pouco suporte para *hot spots*
- **Replicação** de arquivos: suporta arquivos somente leitura, mas não atualizações

Sistemas de arquivos distribuídos - NFS

- Resumo:

- **Heterogeneidade** de hardware e sistema operacional: bem-sucedida
- **Tolerância a falhas**: modelo de falha observado por clientes semelhante ao de arquivos locais (graças à natureza sem estado e idempotente do NFS)
- **Consistência**: muito próxima do modelo “uma cópia”
- **Segurança**: alcançada pela integração do Kerberos
- **Eficiência**: muito boa

Google File System

- Ver material no site