URI | UNIVERSIDADE REGIONAL INTEGRADA DO ALTO URUGUAI E DAS MISSÕES

30-769

Sistemas Operacionais II

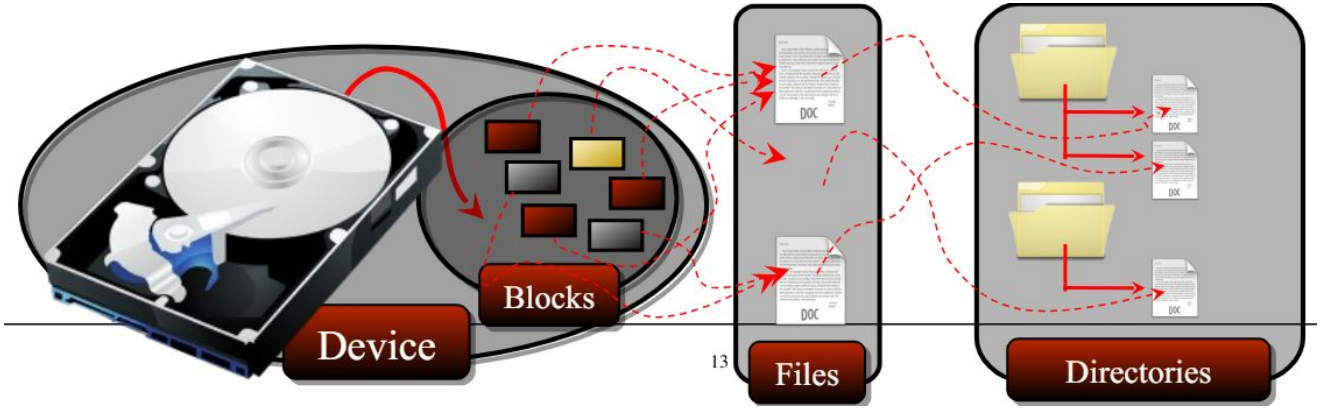
MSc. Fernando Schubert

O QUE É UM SISTEMAS DE ARQUIVOS

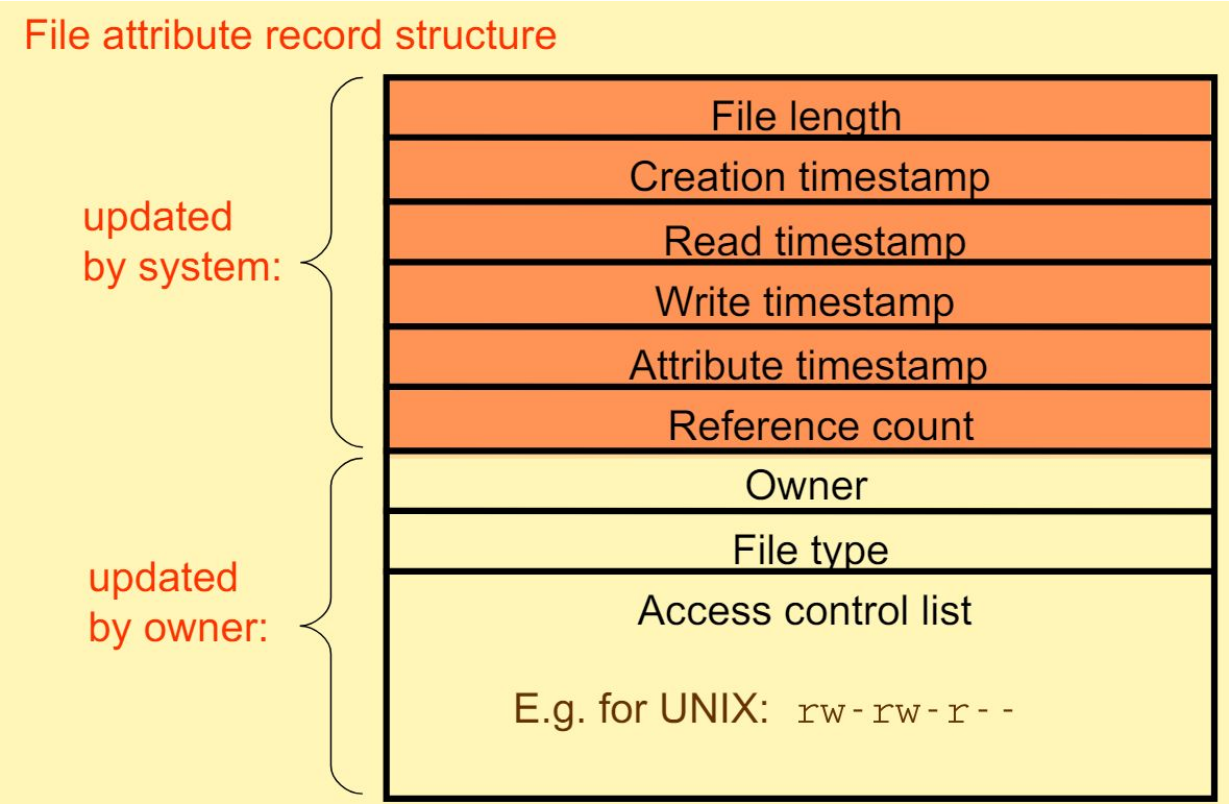
- Conjuntos de dados armazenados de forma persistente
- Espaço de nomes hierárquico visível para todos os processos
- API com as seguintes características:
  - operações de acesso e atualização em conjuntos de dados armazenados de forma persistente
  - Modelo de acesso sequencial (com facilidades adicionais para acesso aleatório)
- Compartilhamento de dados entre usuários, com controle de acesso
- Acesso simultâneo:
  - certamente para acesso somente leitura
  - e quanto às atualizações?
- Outras características:
  - sistemas de arquivos montáveis
  - mais? ...

O QUE É UM SISTEMA DE ARQUIVOS

- **Módulos do sistema de arquivos**
  - **Módulo de diretório**: relaciona nomes de arquivos com IDs de arquivos
  - **Módulo de arquivo**: relaciona IDs de arquivos com arquivos específicos
  - **Módulo de controle de acesso**: verifica a permissão para a operação solicitada
  - **Módulo de acesso a arquivos**: lê ou grava dados ou atributos do arquivo
  - **Módulo de bloco**: acessa e aloca blocos de disco
  - **Módulo de dispositivo**: I/O de disco e buffering



O QUE É UM SISTEMA DE ARQUIVOS



OPERAÇÕES DE SISTEMA DE ARQUIVOS NO UNIX

UNIX file system operations

filedes = open(name, mode) *filedes = creat(name, mode)*	Opens an existing file with the given *name*. Creates a new file with the given *name*. Both operations deliver a file descriptor referencing the open file. The *mode* is *read*, *write* or both.
status = close(filedes)	Closes the open file *filedes*.
count = read(filedes, buffer, n) *count = write(filedes, buffer, n)*	Transfers *n* bytes from the file referenced by *filedes* to *buffer*. Transfers *n* bytes to the file referenced by *filedes* from buffer. Both operations deliver the number of bytes actually transferred and advance the read-write pointer.
pos = lseek(filedes, offset, whence)	Moves the read-write pointer to offset (relative or absolute, depending on *whence*).
status = unlink(name)	Removes the file *name* from the directory structure. If the file has no other names, it is deleted.
status = link(name1, name2)	Adds a new name (*name2*) for a file (*name1*).
status = stat(name, buffer)	Gets the file attributes for file *name* into *buffer*.

SISTEMAS DE ARQUIVOS DISTRIBUÍDOS

Por que precisamos de um sistema de arquivos distribuídos?

- Propósito: conectar usuários e recursos de armazenamento
- Requisitos:
  - Segurança
  - Disponibilidade
  - Redundância
  - Escalabilidade
  - Transparência
  - Eficiência
  - Consistência
  - Tolerância a falhas
  - Heterogeneidade
  - Replicação
  - Concorrência

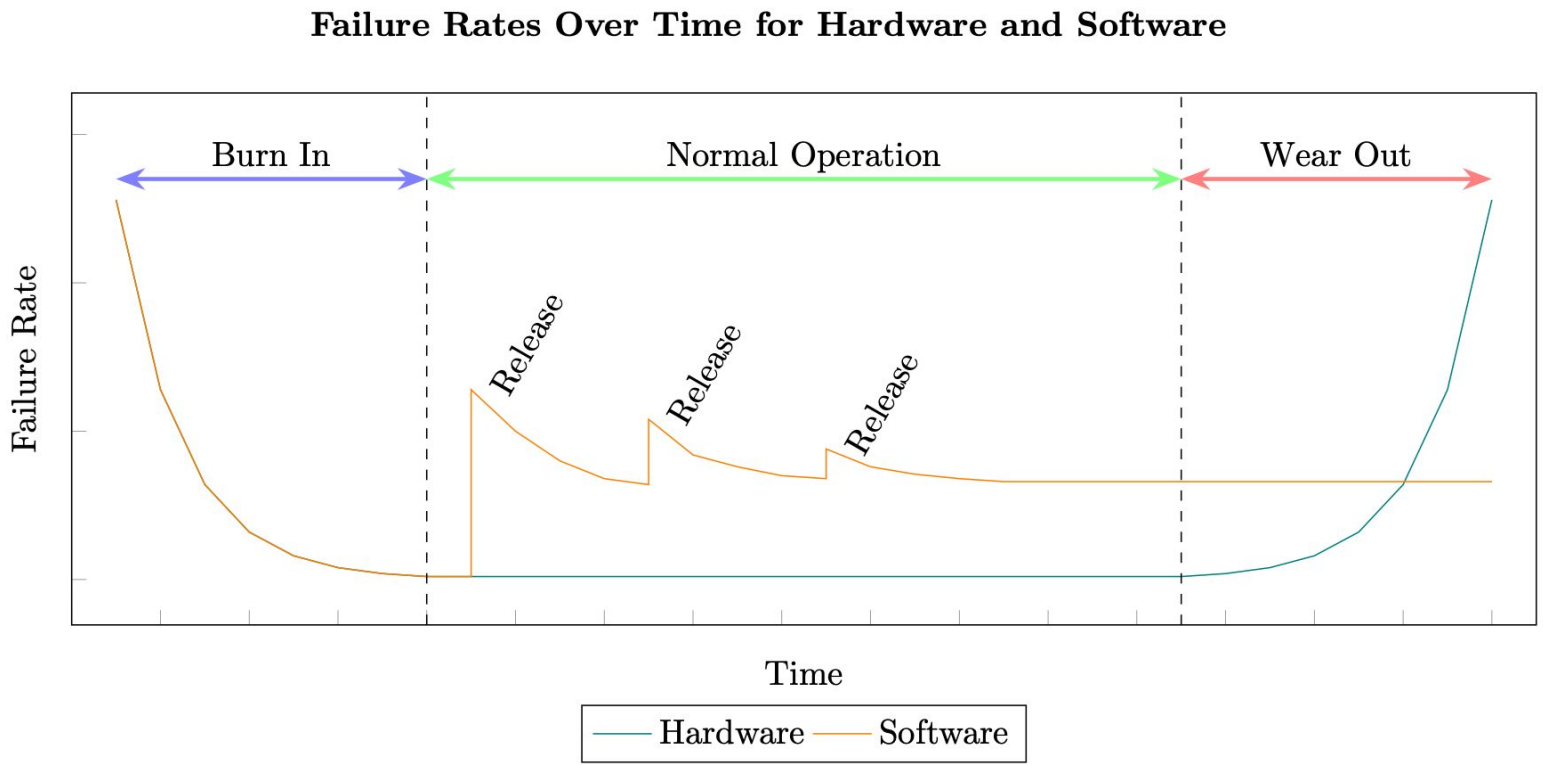
SEGURANÇA

- Mecanismos de controle de acesso baseados em listas de controle
- Necessidade de autenticar as requisições dos clientes
- Controle de acesso baseado nas identidades corretas de usuário
- Proteger conteúdo das mensagens de requisição e respota
    - Assinaturas digitais
    - Criptografia

DISPONIBILIDADE

- Os sistemas distribuídos são compostos por componentes de software e componentes de hardware. Alguns dos componentes do software podem ser, eles próprios, outro sistema distribuído. A disponibilidade dos componentes subjacentes de hardware e software afeta a disponibilidade resultante de sua workload.
- O cálculo da disponibilidade usando MTBF e MTTR tem suas raízes nos sistemas de hardware. No entanto, os sistemas distribuídos falham por motivos muito diferentes dos de um hardware. Quando um fabricante pode calcular consistentemente o tempo médio antes que um componente de hardware se desgaste, o mesmo teste não pode ser aplicado aos componentes de software de um sistema distribuído. O hardware normalmente segue a curva “banhada” da taxa de falhas, enquanto o software segue uma curva escalonada produzida por defeitos adicionais que são introduzidos a cada nova versão.
    - Mean Time Between Failures (MTBF) é a média de tempo entre falhas sucessivas de um sistema ou componente, indicando sua confiabilidade.
    - Mean Time to Repair (MTTR) é a média de tempo necessário para restaurar um sistema ou componente após uma falha, medindo a eficiência da recuperação

DISPONIBILIDADE



TRANSPARÊNCIA

“A transparência é definida como sendo a ocultação, para um usuário final ou para um programador de aplicativos, da separação dos componentes em um sistema distribuído de modo que o sistema seja percebido como um todo, em vez de uma coleção de componentes independentes.” (COULOURIS et al., 2008, p. 34)

TRANSPARÊNCIA

- Implicam principalmente em flexibilidade e escalabilidade
- Equilíbrio com complexidade e desempenho do software
- Transparência de acesso
  - Programas clientes não devem ter conhecimento da distribuição de arquivos
  - Um único conjunto de operações é disponibilizado para acessar arquivos locais e remotos
- Transparência de localização
  - Programas clientes devem ver um espaço de nomes de arquivos uniforme
  - Arquivos podem ser deslocados de um servidor para outro sem alteração de seus nomes de caminho

TRANSPARÊNCIA

- Transparência de mobilidade
  - Nem os programas clientes nem as tabelas de administração de sistema nos nós clientes precisam ser alterados quando os arquivos são movidos
- Transparência de desempenho
  - Programas clientes devem continuar a funcionar satisfatoriamente, mesmo com a oscilação de carga sobre o serviço
- Transparência de escala
  - Serviço pode ser expandido paulatinamente

CONCORRÊNCIA

- Alterações feitas em um arquivo por um cliente não devem interferir na operação de outros clientes que estejam acessando o mesmo arquivo simultaneamente
- Serviços de arquivos atuais são baseados no UNIX
  - Travamento (locking) em nível de arquivo ou de registro
- Exclusão mútua

REPLICAÇÃO DE ARQUIVOS

- Alterações feitas em um arquivo por um cliente não devem interferir na operação de outros clientes que estejam acessando o mesmo arquivo simultaneamente
- Serviços de arquivos atuais são baseados no UNIX
  - Travamento (locking) em nível de arquivo ou de registro
- Exclusão mútua

REPLICAÇÃO DE ARQUIVOS

- Um arquivo pode ser representado por várias cópias de seu conteúdo em diferentes locais
- Vantagens:
  - Permite que vários servidores compartilhem a carga do fornecimento de um serviço para clientes que acessem o mesmo conjunto de arquivos
  - Melhora a tolerância a falhas; clientes podem localizar outro servidor que contenha uma cópia do arquivo
- Há poucos serviços de arquivos que suportem replicação completa
- Caches locais de arquivos (ou partes deles)

TOLERÂNCIA A FALHAS

- É preciso que o sistema de arquivos distribuído continue a funcionar diante de falhas de clientes ou de servidores
- Semântica de invocação "no máximo uma vez"
  - Ou o cliente consegue acessar o arquivo remoto numa única execução, ou uma exceção é lançada
- Semântica de invocação "pelo menos uma vez"
  - Mensagem de requisição é transmitida no caso de a primeira tentativa não obter êxito
- Necessidade de que operações sejam idempotentes
- Servidores podem ser *stateless*
  - Estado anterior à falha não precisa ser recuperado
- Tolerância a falha de desconexão (ou de servidor) exige replicação de arquivo

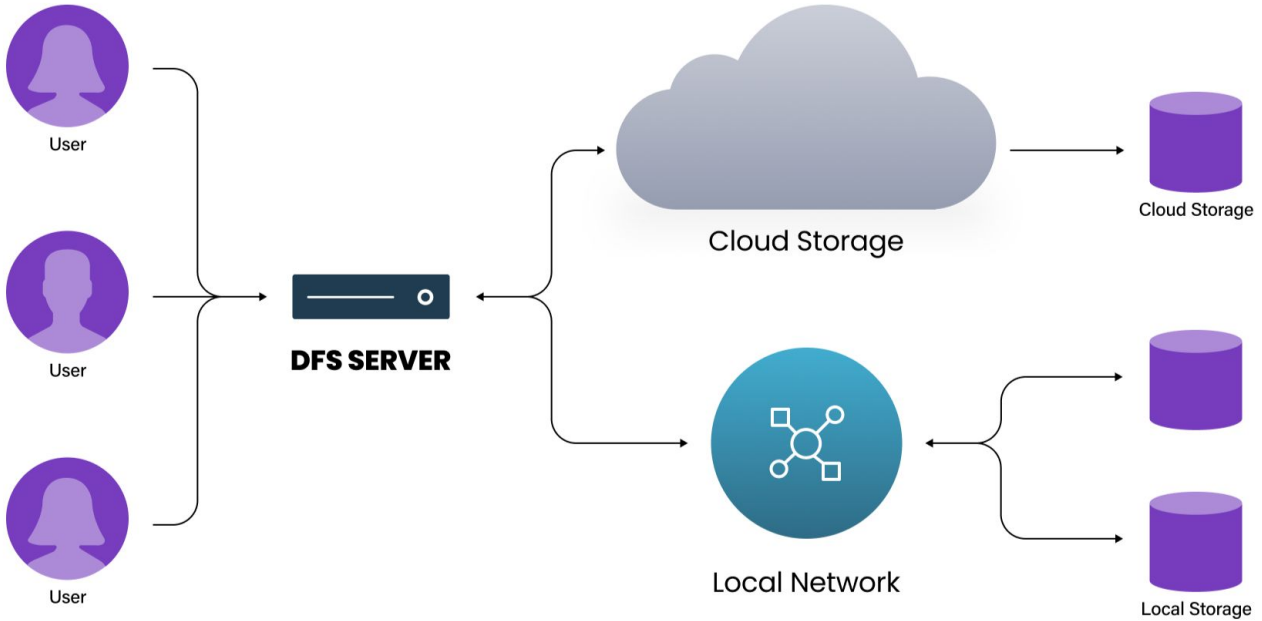
CONSISTÊNCIA

- Em sistemas convencionais, é oferecida a semântica de atualização de cópia única
  - O conteúdo visto por todos é aquele que seria visto se existisse apenas uma única cópia do conteúdo do arquivo
- Em sistemas replicados, ocorrerá algum atraso na propagação das modificações para as cópias existentes
  - Isso acarretará algum desvio da semântica de cópia única

EFICIÊNCIA

- Um serviço de arquivos distribuídos deve oferecer recursos que tenham pelo menos o mesmo poder e generalidade daqueles encontrados nos sistemas de arquivos convencionais e deve obter um nível de desempenho comparável.

VISÃO GERAL


User
User
User
DFS SERVER
Cloud Storage
Cloud Storage
Local Network
Local Storage

EVOLUÇÃO

- Na primeira geração de sistemas distribuídos (1974-95), os sistemas de arquivos (por exemplo, NFS) eram os únicos sistemas de armazenamento em rede.
- Com o advento dos sistemas de objetos distribuídos (CORBA, Java) e da web, o cenário tornou-se mais complexo.
- O foco atual está em armazenamento em grande escala e escalável.
  - Google File System
  - Amazon S3 (Simple Storage Service)
  - Armazenamento em nuvem (por exemplo, DropBox)

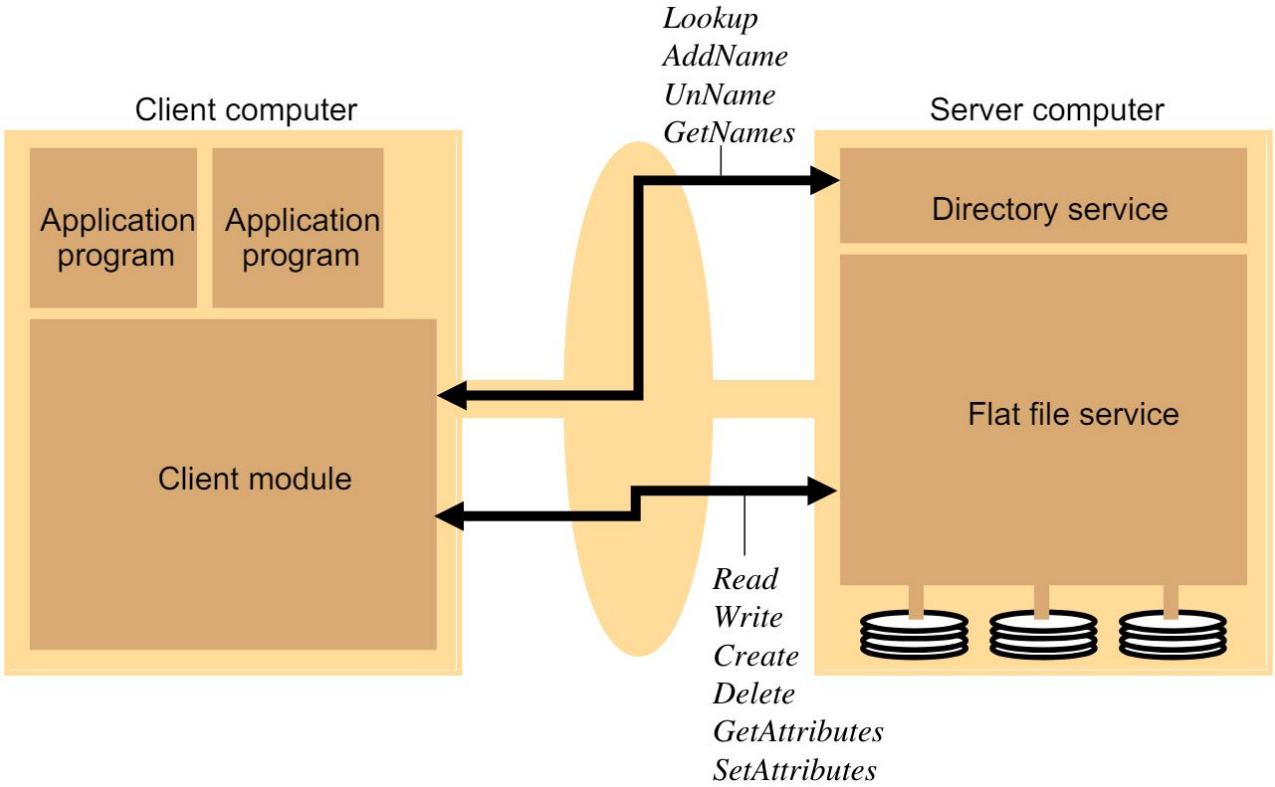
CLASSIFICAÇÃO

- **Sistemas de arquivos distribuídos clássicos**
  a. **NFS**: Sistema de Arquivos de Rede da Sun (Sun Network File System)
  b. **AFS**: Sistema de Arquivos Andrew (Andrew File System)
- **Sistemas de arquivos distribuídos paralelos**

ARQUITETURA

- Uma arquitetura que oferece uma clara separação das principais preocupações ao fornecer acesso a arquivos é obtida ao estruturar o serviço de arquivos em três componentes:
  - Um serviço de arquivos plano
  - Um serviço de diretório
  - Um módulo cliente.
- Os módulos relevantes e suas relações são (mostrados a seguir).
- O módulo cliente implementa as interfaces exportadas pelos serviços de arquivos planos e de diretórios no lado do servidor.

ARQUITETURA



RESPONSABILIDADE DOS MÓDULOS

- Serviço de Arquivos:
  - Envolve a implementação de operações sobre o conteúdo dos arquivos.
  - Identificadores Únicos de Arquivo (UFIDs) são usados para se referir aos arquivos em todas as solicitações de operações do serviço de arquivo plano. UFIDs são longas sequências de bits escolhidas para que cada arquivo seja único entre todos os arquivos em um sistema distribuído.
- Serviço de Diretório:
  - Fornece o mapeamento entre nomes de texto para os arquivos e seus UFIDs. Os clientes podem obter o UFID de um arquivo informando seu nome de texto ao serviço de diretório. O serviço de diretório suporta as funções necessárias para gerar diretórios e adicionar novos arquivos a diretórios.
- Módulo Cliente:
  - É executado em cada computador e fornece um serviço integrado (arquivo plano e diretório) como uma única API para programas aplicativos. Por exemplo, em hosts UNIX, um módulo cliente emula o conjunto completo de operações de arquivos do Unix.
  - Ele armazena informações sobre as localizações na rede dos processos de servidor de arquivos planos e de diretórios; e alcança melhor desempenho por meio da implementação de um cache de blocos de arquivos recentemente usados no cliente.

DFS: ESTUDO DE CASO

NFS (Network File System)

- Desenvolvido pela Sun Microsystems (em 1985)
- Mais popular, aberto e amplamente utilizado.
- O protocolo NFS foi padronizado através do IETF (RFC 1813)

AFS (Andrew File System)

- Desenvolvido pela Universidade Carnegie Mellon como parte dos ambientes de computação distribuída Andrew (em 1986)
- Um projeto de pesquisa para criar um sistema de arquivos em nível de campus.
- Implementação de domínio público disponível no Linux (LinuxAFS)
- Foi adotado como base para o sistema de arquivos DCE/DFS na Open Software Foundation (OSF, www.opengroup.org) DEC (Ambiente de Computação Distribuída)

NFS: ESTUDO DE CASO

Um padrão da indústria para compartilhamento de arquivos em redes locais desde os anos 1980

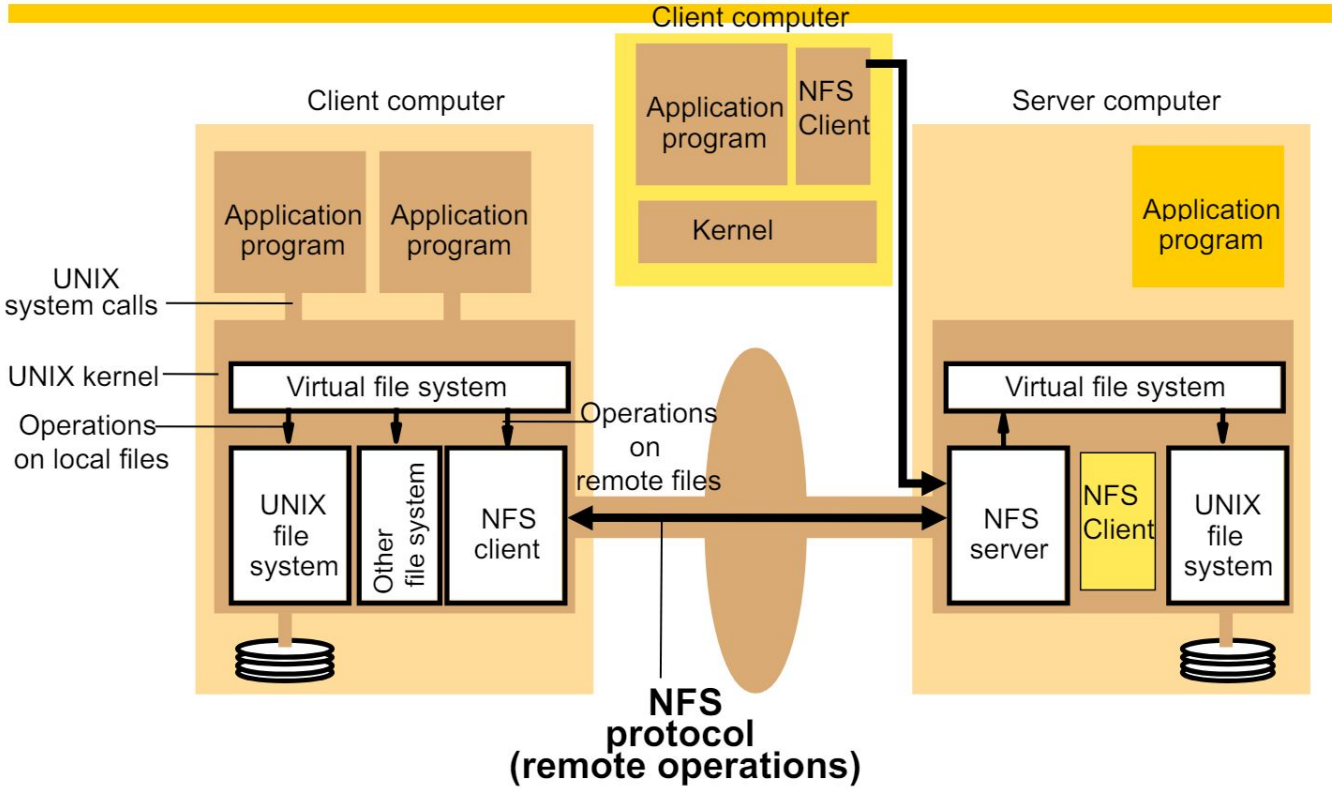
- Um padrão aberto com interfaces claras e simples
- Segue de perto o modelo de serviço de arquivo abstrato definido acima
- Suporta muitos dos requisitos de design já mencionados:
  - transparência
  - heterogeneidade
  - eficiência
  - tolerância a falhas
- Conquista limitada de:
  - concorrência
  - replicação
  - consistência
  - segurança

NFS: ESTUDO DE CASO

1985: Versão Original (uso interno)

- **1989: NFSv2 (RFC 1094)**
  - Operava inteiramente sobre UDP
  - Protocolo sem estado (o núcleo)
  - Suporte para arquivos de 2GB
- **1995: NFSv3 (RFC 1813)**
  - Suporte para 64 bits (> 2GB de arquivos)
  - Suporte para gravações assíncronas
  - Suporte para TCP
  - Suporte para atributos adicionais
  - Outras melhorias
- **2000-2003: NFSv4 (RFC 3010, RFC 3530)**
  - Colaboração com o IETF
  - A Sun transfere o desenvolvimento do NFS
- **2010: NFSv4.1**
  - Adiciona Parallel NFS (pNFS) para acesso paralelo a dados
- **2015**
  - RFC 7530

NFS: ESTUDO DE CASO



NFS: ESTUDO DE CASO

Controle de acesso e autenticação do NFS

- Servidor sem estado, portanto a identidade do usuário e os direitos de acesso devem ser verificados pelo servidor em cada solicitação.
  - No sistema de arquivos local, eles são verificados apenas no open()
- Cada solicitação do cliente é acompanhada pelo userID e groupID
  - que são inseridos pelo sistema RPC
- O servidor fica exposto a ataques de impostores, a menos que o userID e o groupID sejam protegidos por criptografia
- O Kerberos foi integrado ao NFS para fornecer uma solução de segurança mais forte e abrangente

NFS: ESTUDO DE CASO

Servidor:

- **nfsd**: Daemon do servidor NFS que atende às solicitações dos clientes.
- **mountd**: Daemon de montagem NFS que executa a solicitação de montagem passada pelo nfsd.
- **rpcbind**: Mapeador de portas RPC usado para localizar o daemon nfsd.
- **/etc/exports**: Arquivo de configuração que define quais partes dos sistemas de arquivos são exportadas através do NFS e como.

Cliente:

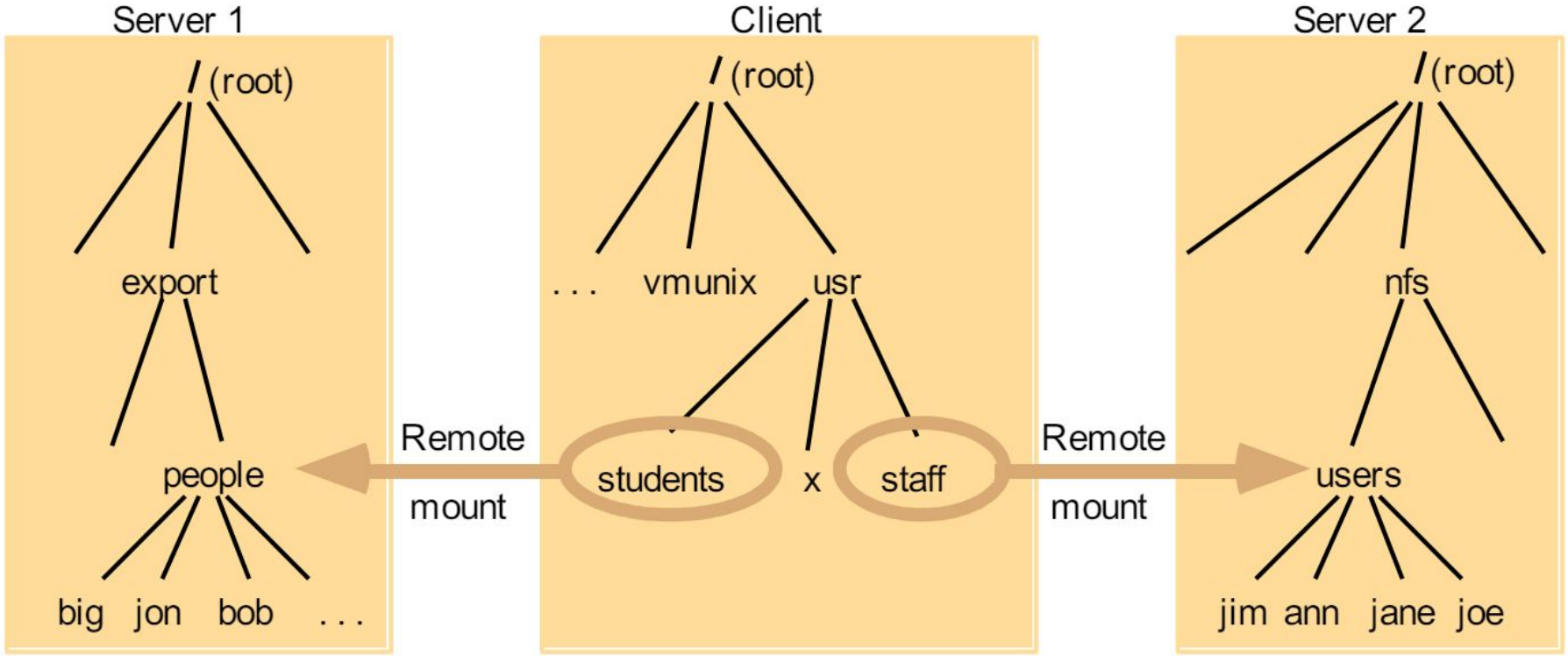
- **mount**: Comando padrão de montagem de sistema de arquivos.
- **/etc/fstab**: Arquivo de tabela de sistemas de arquivos.
- **nfsiod**: (opcional) Servidor local de E/S NFS assíncrono.

NFS: ESTUDO DE CASO

Serviço de Montagem

- Operação de montagem:
  `mount(remotehost, remotedirectory, localdirectory)`
- O servidor mantém uma tabela de clientes que montaram sistemas de arquivos nesse servidor
- Cada cliente mantém uma tabela de sistemas de arquivos montados contendo:
  `<endereço IP, número da porta, identificador de arquivo>`
- Montagens "hard" versus "soft"
-

NFS: ESTUDO DE CASO



NFS: ESTUDO DE CASO

Serviço de Montagem

- Operação de montagem:
  `mount(remotehost, remotedirectory, localdirectory)`
- O servidor mantém uma tabela de clientes que montaram sistemas de arquivos nesse servidor
- Cada cliente mantém uma tabela de sistemas de arquivos montados contendo:
  `<endereço IP, número da porta, identificador de arquivo>`
- Montagens "hard" versus "soft"
-

ATIVIDADE I

ARTIGO: https://www.sbc.org.br/documentos-da-sbc/category/169-templates-para-artigos-e-capitulos-de-livros

Cada aluno irá pequisar sobre um Sistema de Arquivos Distribuídos com o seguinte objetivo:

- Introdução - definir o que é um sistema de arquivos distribuídos, principais características
- Apresentar o SAD alvo da pesquisa abordando:
  - Como os servidores armazenam os dados
    - Particionamento e replicação
    - Segurança
    - Tolerância a falhas
    - Transparência
    - Nomeação
    - Como os dados são armazenados (arquivos, blocos)
    - Caching e política de trocas
  - Como os clientes acessam os dados
  - Popularidade