# Sistemas de Arquivos Distribuídos

**Prof. Dr. João Carlos Lopes Fernandes**
E-mail: joao.fernandes1@docente.unip.br

# Objetivo

- Permitir que programas armazenem e acessem arquivos remotos exatamente como se fossem locais
  - Possibilitando que usuários acessem arquivos a partir de qualquer computador em uma rede
- Capta a essência do compartilhamento de recursos pregado pelos desenvolvedores de sistemas distribuídos
- Trataremos aqui do serviço básico de sistemas de arquivos
  - Neste momento, não há o tratamento de outras questões, como replicação, garantia de largura de banda ou transmissão multimídia em tempo real

# Sistemas de arquivos "tradicionais"...

- Originalmente desenvolvidos para computadores centralizados e desktop
- Recurso do sistema operacional que fornece uma interface de programação conveniente para armazenamento em disco
  - Controle de acesso
  - Mecanismos de proteção
  - Úteis para o compartilhamento de dados e programas
- Sistemas de arquivos distribuídos suportam compartilhamento de informações em toda a rede

# Características dos sistemas de arquivos (1)

- Responsáveis pela organização, armazenamento, recuperação, atribuição de nomes, compartilhamento e proteção de arquivos
  - Programadores não têm que se preocupar com detalhes de alocação e leiaute de armazenamento físico no disco
- Arquivos contém dados e atributos:
  - Dados são uma seqüência de elementos acessíveis pelas operações de leitura e escrita de qualquer parte deste seqüência
  - Atributos são mantidos como um único registro, contendo informações sobre o arquivo
    - Tamanho do arquivo
    - Identidade do proprietário
    - Listas de controle de acesso

# Características dos sistemas de arquivos (2)

- Atributos escondidos (*shadow attributes*)
  - São gerenciados pelo próprio sistema de arquivos
  - Normalmente não podem ser atualizados por programas de usuário

| Tamanho do Arquivo |
| --- |
| Horário de Criação |
| Horário de Acesso (Leitura) |
| Horário de Modificação (Escrita) |
| Horário de Alteração de Atributo |
| Contagem de Referência |
| Proprietário |
| Tipo de Arquivo |
| Lista de Controle de Acesso |

# Diretórios (1)

- Sistemas de arquivos são projetados para armazenar e gerenciar um grande número de arquivos
- Recursos:
  - Criação
  - Atribuição de Nomes
  - Exclusão de Arquivos
- Atribuição de nomes de arquivos é suportada pelo uso de diretórios

# Diretórios (2)

- Trata-se de um arquivo, freqüentemente de um tipo especial
- Fornece um mapeamento dos nomes textuais para identificadores internos de arquivo
- Podem incluir nomes de outros diretórios
  - Atribuição hierárquica de nomes
  - Nomes de caminho (*pathname*)

# Controle de acesso

- Sistemas de arquivos também assumem a responsabilidade pelo controle de acesso aos arquivos
- Restrições de acesso de acordo com as autorizações dos usuários e com o tipo de acesso solicitado
  - Leitura
  - Escrita/atualização
  - Execução
  - ...

# Metadados

- Termo utilizado para se referir às informações extras armazenadas por um sistema de arquivos
- Conjunto de informações padronizadas sobre os arquivos
  - Representam a criação e a remoção de arquivos e diretórios, ampliação de um arquivo, um truncamento de um arquivo, etc.
  - Necessários para fazer o gerenciamento dos arquivos dentro do disco rígido

# Módulos do Sistema de Arquivos (1)

- Estrutura em camadas para implementação de um sistema de arquivos típico
  - Não-distribuído
  - Sistema operacional convencional
- Cada camada depende apenas das camadas que estão abaixo dela
  - Implementação de um serviço de arquivos precisa de todos estes componentes
  - Componentes adicionais são utilizados para tratar da comunicação cliente-servidor e da atribuição de nomes e da localização de arquivos distribuídos


Prof. Dr. João Carlos Lopes Fernandes
E-mail: joao.fernandes1@docente.unip.br

# Módulos do Sistema de Arquivos (2)

- Módulo de diretório
  - Relaciona nomes de arquivos com IDs de arquivos
- Módulo de arquivo
  - Relaciona IDs de arquivos com arquivos em particular
- Módulo de controle de acesso
  - Verifica a permissão para a operação solicitada
- Módulo de acesso a arquivo
  - Lê ou escreve dados, ou atributos
- Módulo de bloco
  - Acessa e aloca blocos de disco
- Módulo de dispositivo
  - I/O de disco e uso de *buffers*

# Operações de Sistemas de Arquivos (1)

- Tratam-se de chamadas de sistema implementadas pelo kernel
- Programadores de aplicativos as acessam por meio de funções em bibliotecas. Exemplos:
  - Biblioteca de entrada e saída padrão da linguagem C
  - Classes para manipulação de arquivos em Java
- Primitivas servem como indicação das operações que os sistemas de arquivos devem suportar
  - As descritas a seguir estão disponíveis nos sistemas UNIX, mas podem servir de referência para outros sistemas de arquivos


Prof. Dr. João Carlos Lopes Fernandes
E-mail: joao.fernandes1@docente.unip.br

# Operações de Sistemas de Arquivos (2)

- filedes = open(nome, modo)
  - Abre um arquivo existente com o nome fornecido
  - 'Modo' pode ser *read*, *write* ou ambos
- filedes = creat(nome, modo)
  - Cria um novo arquivo com o nome fornecido
  - Estas duas operações produzem um descritor de arquivo referenciando o arquivo aberto
- status = close(filedes)
  - Fecha o arquivo referenciado por *filedes*

# Operações de Sistemas de Arquivos (3)

- count = read(filedes, buffer, n)
  - Transfere para o buffer *n* bytes do arquivo referenciado por *filedes*
- count = write(filedes, buffer, n)
  - Transfere *n* bytes de buffer para o arquivo referenciado por *filedes*
  - Estas duas operações retornam o número de bytes realmente transferidos e avançam o ponteiro de leitura e escrita
- pos = lseek(filedes, offset, whence)
  - Desloca o ponteiro de leitura e escrita para *offset* (relativo ou absoluto), dependendo do *whence*
    - SEEK_SET: deslocamento deve ser de *offset* bytes
    - SEEK_CUR: deslocamento deve ser de *offset* bytes a partir da posição atual
    - SEEK_END: deslocamento deve ser de *offset* bytes a partir do final do arquivo

# Operações de Sistemas de Arquivos (4)

- status = unlink(nome)
  - Remove o arquivo *nome* da estrutura de diretórios
  - Se o arquivo não tiver outras referências, ele será excluído
- status = link(nome1, nome2)
  - Cria uma nova referência, ou nome (*nome2*), para um arquivo (*nome1*)
- status = stat(nome, buffer)
  - Escreve os atributos do arquivo *nome* em *buffer*


Prof. Dr. João Carlos Lopes Fernandes
E-mail: joao.fernandes1@docente.unip.br

# Requisitos do Sistema de Arquivos Distribuído

- Desenvolvimento dos primeiros sistemas de arquivos distribuídos detectaram os primeiros requisitos e armadilhas dos projetos de sistemas distribuídos
- Os primeiros ofereciam transparência de acesso e de localização
  - Depois, surgiu a demanda pelo oferecimento de desempenho, escalabilidade, concorrência, tolerância a falhas e segurança...


Prof. Dr. João Carlos Lopes Fernandes
E-mail: joao.fernandes1@docente.unip.br

# Transparência (1)

- 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 (2)

- 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

# Controle de 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

- 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)


Prof. Dr. João Carlos Lopes Fernandes
E-mail: joao.fernandes1@docente.unip.br

# Diferentes hardwares e Sistemas Operacionais

- Interfaces do serviço de arquivos distribuídos deve ser implementado de forma que o software cliente e servidor possa ser implementado para diferentes sistemas operacionais e computadores


Prof. Dr. João Carlos Lopes Fernandes
E-mail: joao.fernandes1@docente.unip.br

# 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 na primeira tentativa não obter êxito
  - Necessidades 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 arquivos

# Consistência

- Em sistemas convencionais, é oferecida a semântica de atualização de cópia única
  - 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
  - Acarretará algum desvio da semântica de cópia única

# 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

# Eficiência

- Sistemas de arquivo distribuídos devem prover a mesma generalidade dos serviços convencionais
- Desempenho deverá também se equiparar

"Desejaríamos ter um servidor de arquivos simples, de baixo nível, para compartilhar um recurso dispendioso, a saber, o disco, para que nos deixasse livres para projetar um sistema de arquivos mais apropriado a um cliente em particular, ao mesmo tempo em que disponibilizasse um sistema de alto nível compartilhado entre os clientes"

# Arquitetura de Sistemas de Arquivos Distribuídos (1)

- Sistema de Arquivos é estruturado com três componentes principais:
  - Serviço de arquivos planos
  - Serviço de diretórios
  - Módulo cliente
- Dois primeiros exportam uma interface para uso pelos programas clientes
  - Interface RPC que provê um conjunto compreensivo de operações de acesso aos arquivos
- Módulo cliente provê uma interface de programação simples com operações de acesso similares àquelas encontradas nos sistemas de arquivos convencionais

# Arquitetura de Sistemas de Arquivos Distribuídos (2)

- Design é considerado aberto
  - Diferentes módulos clientes podem ser utilizados para implementar diferentes interfaces de programação
  - Simulação de operações de arquivos de diferentes sistemas operacionais
  - Otimização de performance para diferentes configurações de hardware de clientes e servidores

# Serviço de arquivos planos

- Implementação de operações sobre o conteúdo dos arquivos
- Utilização de UFIDs (Identificadores Únicos de Arquivos)
  - Longas seqüências de bits escolhidos de tal forma que cada arquivo tenha um identificador único dentro do sistema distribuído
  - Utilizados para referenciar arquivos em todas as requisições por operações de arquivos
- Quando um serviço de arquivos planos recebe uma requisição para criar um arquivo, ele gera um novo UFID e o retorna para o solicitante
- Divisão de responsabilidades entre o serviços de arquivos e de diretórios é baseado na forma de uso do UFID

# Serviço de diretórios

- Provê um mapeamento entre *nomes textuais* para arquivos e seus respectivos UFIDs
  - Clientes podem obter o UFID de um arquivo acenando com seu nome textual para um serviço de diretórios
- Oferecem as funções necessárias para:
  - Gerar diretórios
  - Adicionar novos nomes de arquivos para diretórios
  - Obter UFIDs desses diretórios
- Funciona como o cliente de um serviço de arquivos planos
  - Arquivos de diretórios são armazenados em arquivos do serviço de arquivos
  - Quando esquemas hierárquicos são adotados, diretórios mantém referências para outros diretórios

# Módulo Cliente

- Integra e estende operações dos serviços de arquivos e diretórios do lado do cliente
  - Deve executar em qualquer computador cliente
- Interface de programação disponível para programas em nível de usuário
- Mantém informações sobre a localização de rede dos servidores de arquivos e de diretórios
- Pode manter *cache* de blocos de arquivos recentemente utilizados no cliente

# Interface de serviço de arquivos planos (1)

- Interface RPC utilizada pelo cliente para acessar o serviço
- Normalmente utilizada diretamente pelo aplicativo em nível de usuário
- Read(FileId, i, n) ⟶ Data
  - Lê uma seqüência de *n* itens de um arquivo iniciando da posição i
  - Retorna os dados resultantes da operação
  - Um *FileId* (UFID) será inválido se:
    - Referido arquivo não está presente no servidor processando a requisição, ou
    - Se as permissões de acesso forem inapropriadas para a operação requisitada
  - 1≤ i ≤ Length(File)
    - Caso contrário, *throws BadPosition*

# Interface de serviço de arquivos planos (2)

- Write(FileId, i, Data)
  - Escreve uma seqüência de *Data* em um arquivo iniciando da posição i, estendendo o arquivo se necessário
  - Se for o caso, realoca o conteúdo anterior do arquivo
  - $1 \leq i \leq \text{Length(File)}+1$
    - Caso contrário, *throws BadPosition*
- Create() ⟶ FileId
  - Cria um novo arquivo de tamanho 0 e retorna um UFID para ele
- Delete(FileId)
  - Remove um arquivo

# Interface de serviço de arquivos planos (3)

- GetAttributes(FileId) ⟶ Attr
  - Retorna os atributos do arquivo
- SetAttributes(FileId, Attr)
  - Modifica os atributos do arquivo
  - Não pode modificar atributos escondidos

# Controle de acesso (1)

- Em sistemas de arquivos locais, direitos de acesso do usuário são confrontados com o modo de acesso requisitado na operação *open* (*r* ou *w*)
- Arquivo é aberto somente se o usuário tem os direitos necessários
- Identidade do usuário (UID) usada na checagem dos direitos de acesso é o resultado da operação anterior de login autenticado
- Direito de acesso é mantido até o arquivo ser fechado
  - Nenhuma checagem adicional é solicitada quando operações subseqüentes sobre o mesmo arquivo são solicitadas

# Controle de acesso (2)

- Em implementações distribuídas, a checagem de direitos de acesso tem que ser realizada no servidor
  - Interface RPC do servidor é um ponto de acesso desprotegido para os arquivos
- Um ID de usuário tem que ser passado com as solicitações
  - Servidor é vulnerável a identidades forjadas
- Se o resultado da checagem de direito de acesso for mantido no servidor para requisições posteriores, ele deixaria de ser *stateless*

# Controle de acesso (3)

- Alternativas à retenção de autenticação
  - Checagem de acesso é feita sempre que um nome de arquivo for convertido para um UFID.
  - Uma identificação de usuário é submetida a cada requisição de cliente e checagem de acesso é feita pelo servidor para cada operação de arquivo


Prof. Dr. João Carlos Lopes Fernandes
E-mail: joao.fernandes1@docente.unip.br

# Interface de serviço de diretórios (1)

- Propósito primário é oferecer um serviço para traduzir nomes textuais para UFIDs
- Para isso, mantém arquivos de diretórios contendo o mapeamento entre nomes textuais e UFIDs
- Cada diretório é armazenado como um arquivo convencional, com seu próprio UFID
  - Serviço de diretório é um cliente do serviço de arquivos

# Interface de serviço de diretórios (2)

- Lookup(Dir, Name) ⟶ FileId
  - Localiza o nome texto no diretório e retorna o UFID
    - Tradução básica *Name* ⟶ *UFID*
  - Se *Name* não está no diretório *Dir*, exceção *NotFound* é lançada
- AddName(Dir, Name, File)
  - Se *Name* não está no diretório, adiciona (*Name*, *File*) ao diretório e atualiza o registro de atributos do arquivo
  - Se *Name* já existe dentro do diretório, exceção *NameDuplicated* é lançada

# Interface de serviço de diretórios (3)

- UnName(Dir, Name)
  - Se *Name* está no diretório, a entrada contendo *Name* é removida do diretório
  - Se *Name* não está no diretório, excecão NotFound é lançada
- GetNames(Dir, Pattern) $\longrightarrow$ NameSeq
  - Retorna todos os nomes texto no diretório que atende à expressão regular Pattern


Prof. Dr. João Carlos Lopes Fernandes
E-mail: joao.fernandes1@docente.unip.br

# Sistema de arquivos hierárquico

- Trata-se de um conjunto de diretórios arrumados numa estrutura em árvore
- Cada diretório mantém os nomes dos arquivos e de outros diretórios que sejam acessíveis através dele
- Cada diretório pode ser referenciado por um *pathname*
  - Um nome muti-partes que representa um caminho através da árvore
  - A raiz tem um nome distinto e cada arquivo ou diretório tem um nome no diretório
- Arquivos podem ter diversos nomes, e eles podem estar no mesmo ou em diretórios diferentes
  - No UNIX, operação *link*

# Sun Network File System (NFS)

- Originalmente desenvolvido pela Sun Microsystems em 1984
- Permite que um usuário sobre um computador cliente acesse arquivos em uma rede de maneira similar a como o armazenamento local é acessado
- Baseado na especificação RFC1094
  - Originalmente baseado no protocolo UDP
- Expansões posteriores
  - RFC 1813: suporte para acesso via TCP
    - Atualmente é compatível com ambos
  - RFC 3010 e 3530:
    - Aprimoramento de performance e segurança
    - Protocolo stateful

# Sun Network File System (NFS)

- Independente de sistema operacional, mas foi inicialmente desenvolvido para UNIX
- Módulo servidor reside no kernel de cada computador que atua como servidor
- Requisições por arquivos em sistemas de arquivos remotos são traduzidos por um módulo cliente para operações do protocolo NFS
  - São então passadas para o módulo servidor NFS

# Sistema de arquivos virtual (VFS)

- Mecanismo pelo qual o NFS provê transparência de acesso
  - Programas de usuário podem requisitar operações de arquivo para sistemas locais ou remotos sem distinção
- Pode ser adicionado ao kernel do SO para fazer a distinção entre arquivos locais e remotos
  - Faz também a tradução entre os identificadores de arquivos utilizados pelo NFS e aqueles utilizados localmente


Prof. Dr. João Carlos Lopes Fernandes
E-mail: joao.fernandes1@docente.unip.br

# Sistema de arquivos virtual (VFS)

- Identificadores de arquivos utilizados pelo NFS são denominados *handlers*
  - Contém quaisquer informações que o servidor necessite para distinguir um arquivo individualmente
- Campos do identificador:
  - Identificador do sistema de arquivos
  - Número do arquivo *i-node*
  - Número de geração do *i-node*

# Sistema de arquivos virtual (VFS)

- Identificador do sistema de arquivos
  - Número único que é alocado para cada sistema de arquivos quando ele é criado
- *i-node* do arquivo
  - Número que serve para identificar e localizar o arquivo dentro do sistema de arquivos em que ele esteja armazenado
- Número de geração do *i-node*
  - Necessário para identificar unicamente os arquivos quando são criados, já que o número de arquivo *i-node* é reusado quando um arquivo é removido
  - É armazenado com cada arquivo e é incrementado cada vez que um número de arquivo *i-node* é reusado