Network File System

- Network File System é um sistema que permite a montagem de sistemas de arquivos remotos em uma rede TCP/IP
- Desenvolvido pela SUN nos anos 80
- Baseado na especificação RFC1094
  - Originalmente baseado no protocolo UDP
- Extensões posteriores (RFC 1913)
  - Especifica uso com o TCP
- Assim, computadores em uma rede local podem compartilhar seus sistemas de arquivos como se fosse um único sistema de arquivos global
- Acesso a arquivos remotos é idêntico ao acesso a arquivos locais

1

Terminologia do NFS

- Servidor NFS
  - Um servidor de arquivos NFS determina os sistemas de arquivos locais que serão compartilhados com outras máquinas
- Cliente NFS
  - Um cliente NFS monta os sistemas de arquivos compartilhados através da rede e os trata como se fossem locais
- Cliente e servidor se comunicam através de RPC (Chamada Remota de Procedimentos)

NFS lida com heterogeniedade

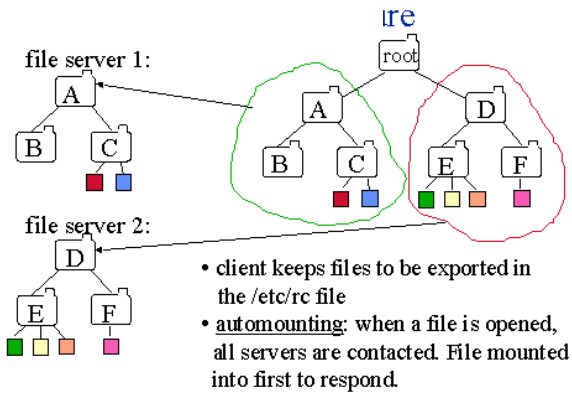
- Foi projetado para ser usado para diferentes sistemas operacionais e arquiteturas de rede (muito comum em uma LAN)
- O protocolo usa Remote Procedure Call (RPC) e uma External Data Representation (XDR), que provêm interfaces independentes de implementação (sistemas e linguagens)

3

Benefícios do NFS

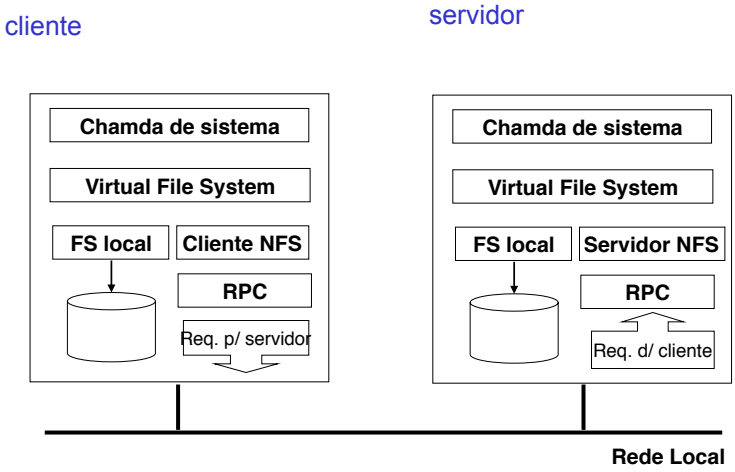
- Arquivos compartilhados pela rede
  - Os arquivos estão localizados no servidor
  - Ficam disponíveis para vários usuários simultaneamente
- Softwares comuns
  - Pacotes de software podem ser compartilhados
  - Diminui o espaço gasto em disco e facilita a gerência.
- Os arquivos remotos parecem ser locais
  - A distribuição de arquivos é transparente para o usuário e as aplicações
  - Arquivos remotos são montados na árvore de diretórios local

Sistema de Arquivos Distribuído através do NFS

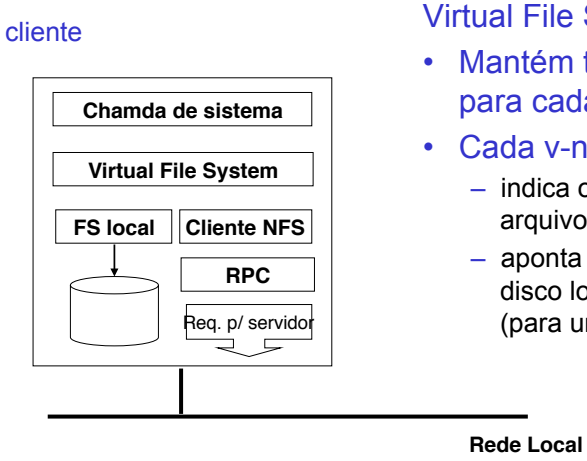


- servidor disponibiliza (exporta) o sistema através do arquivo
- /etc/exports
- Cliente precisa montá-lo através do comando mount ou com uma entrada do arquivo /etc/fstab

Arquitetura do NFS



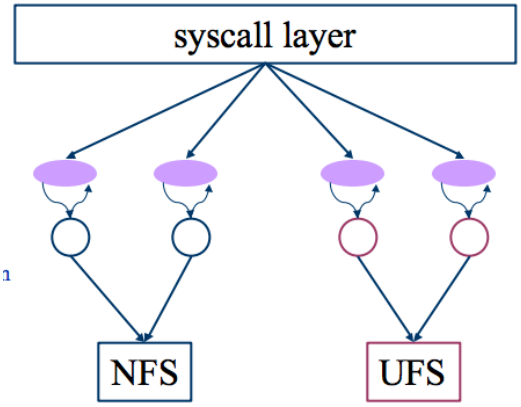
Funcionamento do NFS



Virtual File System:

- Mantém tabela de v-nodes para cada arquivo aberto
- Cada v-node:
  - indica o tipo de sistema de arquivo e
  - aponta ou para um i-node do disco local ou para um r-node (para um arquivo remoto)

Virtual File System / vnodes



Cada arquivo e diretório em uso é representado por um objeto na memória do núcleo: *vnode*.

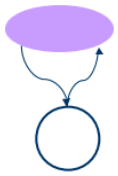
É um struct padrão com atributos do arquivo/dir.

Cada FS específico mantém cache de *vnodes*

9

Vnodes: operações e atributos

vnode attributes (*vattr*)
type (VREG, VDIR, VLNK, etc.)
mode (9+ bits of permissions)
nlink (hard link count)
owner user ID
owner group ID
filesystem ID
unique file ID
file size (bytes and blocks)
access time
modify time
generation number



directories only
vop_lookup (OUT vpp, name)
vop_create (OUT vpp, name, vattr)
vop_remove (vp, name)
vop_link (vp, name)
vop_rename (vp, name, tdvp, tvp, name)
vop_mkdir (OUT vpp, name, vattr)
vop_rmdir (vp, name)
vop_symlink (OUT vpp, name, vattr, contents)
vop_readdir (uio, cookie)
vop_readlink (uio)

generic operations
vop_getattr (vattr)
vop_setattr (vattr)
vhold()
vholdrele()

files only
vop_getpages (page**, count, offset)
vop_putpages (page**, count, sync, offset)
vop_fsync ()

- Operações em vnodes são mapeadas para operações correspondentes nos FS específicos.
- Protocolo NFS possui uma operação para cada operacão em vnodes, com tipos similares de argumentos e resultados.

10

Percorrendo pathnames

- Cada pathname (“/usr/local/abc”) precisa obter o v-node (do objeto “abc”)
- Isso é feito por uma sequência de vop_lookup que descem a árvore de *vnode*

Exemplo:

```
open (“/usr/local/abc”)
vp = get vnode de / (root)
vp->vop_lookup(&cvp, “usr”);
vp = cvp;
vp->vop_lookup(&cvp, “local”);
vp = cvp;
vp->vop_lookup(&cvp, “abc”);
```

… considerando:
- mounting points
- Obtendo vnode do rood ou durrend dir
- Guardando em cache traduções “nome” -> vnode
- Resolvendo soft symbolic links

11

Protocolo NFS

Montagem de um volume remoto

- Através do comando mount:

```
mount servidor.com.br:/misc/export /misc/local
```

- Adicionar entrada em /etc/fstab, indicando o servidor, co path e o mount point local

```
server:/usr/local/pub      /pub   nfs  rsize=..,wsize=..,…
```

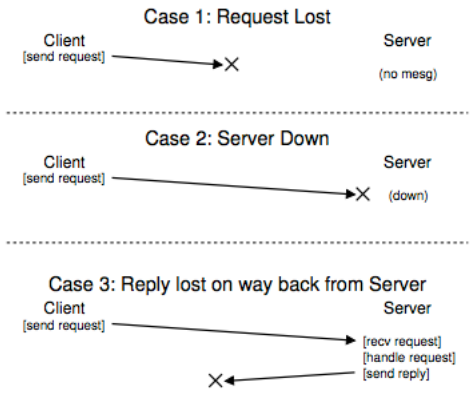
- Usando autofs (o automout daemon, só faz o mount quando houver o acesso ao arquivo ou diretório)
  - Arquivo `/etc/auto.master`: para cada mount point há um arquivo de map (map file), que indica o diretório remoto e as opções de montagem

- O mount() no cliente gera um RPC MOUNT REQUEST para servidor, que é respondido com um RPC MOUNT REPLY (que retorna um file handle do diretório/volume montado)

12

Protocolos NFS

- NFS é stateless (não é mantida qualquer informação dos clientes no servidor)
  - Não existe uma requisição do cliente para abrir um arquivo (não suporta open/close), senão servidor precisaria saber quais estão abertos (e em qual posição)
  - As requisições LOOKUP, READ e WRITE são idempotentes (contém todas as informaçoes necessárias)



Se houver falha, cliente nãoareceberá a confirmação e precisará repetir a operacão

Protocolos NFS

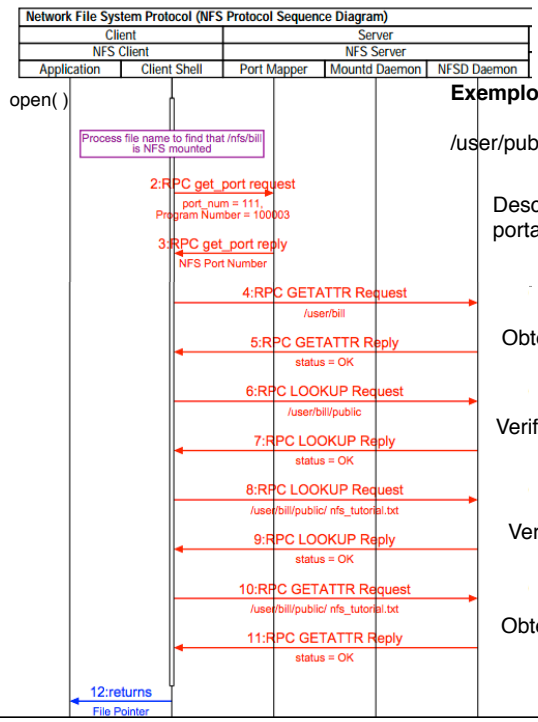
Protocolo de Acesso ao Arquivo:

Requisições GETATTR e LOOKUP

- Cliente faz o parse hierarquico do caminho fazendo várias requisições de GETATTR e LOOKUP
- Se o caminho indicado é válido e o diretório tiver sido exportado pelo servidor, então o servidor responde com um handle

  file handle = (file system type, volumeID, # i-node, inf.de segurança)

- O handle é um token de 32 bits que inclui todas as informações necessárias para identificar o arquivo/objeto no servidor e obter um ponteiro para ele.
- Cliente NFS cria um v-node apontando para um r-node



Exemplo:

/user/public/bill/nfs_tutorial.txt montado em nfs/bill

Descobre IP servidor e faz consulta sobre a porta de acesso ao nfsd

Obtém os atributos de arquivo para /user/bill

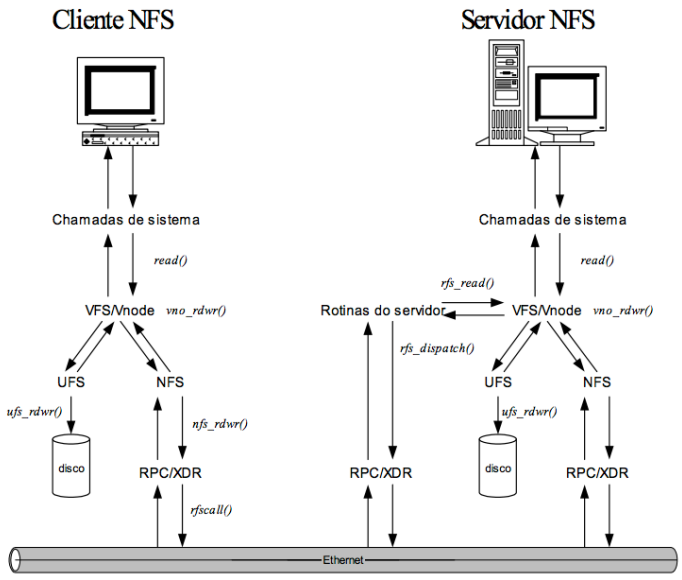
Verifica se subdiretorio public está disponível

Verifica arquivo procurado existe

Obtém os atributos do arquivo requisitado

15

Leitura/Escrita Remota



Protocolo NFS: Leitura

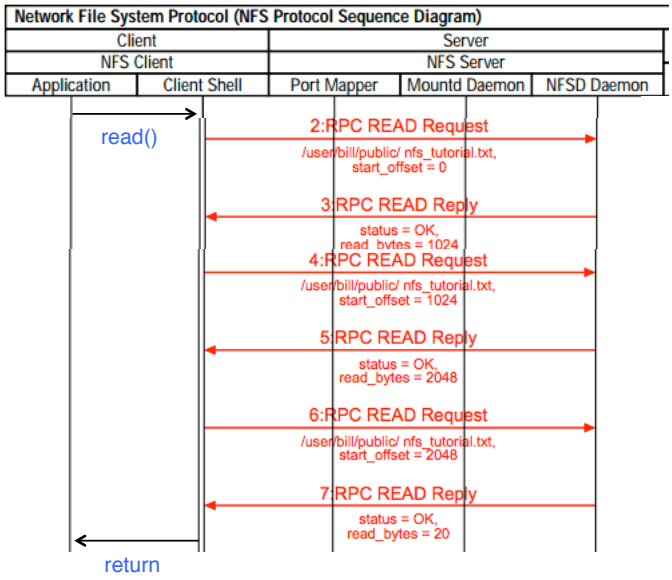
Cliente acessa um arquivo em um servidor NFS:

- Consulta o v-node local e obtém o file handle
- Requisições de leitura/escrita carregam o file handle, e o offset
- Executa vários READ RPC para o servidor
- Recebe como resposta os blocos
- Faz o cache dos blocos

O Servidor NFS:

- Recebe a requisição RPC, que é direcionada para o rpc.nfsd
- Se o v-node, aponta para i-node, atende localmente a requisição fazendo acesso ao disco local
- Se v-node aponta para r-node, faz-se uma requisição para o servidor correto (como se fosse um cliente)

17



18

Caching

- Clientes guardam blocos de dados e diretórios em uso no cache (para evitar latência do RPC na rede)
- Se cliente A têm um bloco de um arquivo remoto F em seu cache, e cliente B modifica esse arquivo F, não há notificação do servidor para A invalidar/atualizar o bloco no seu cache
- Cada bloco vem com um timestamp da última modificação do arquivo no servidor. Após um invalidation_period, antes de buscar um novo bloco (ou na re-abertura do arquivo), o timestamp é comparado com o do servidor, e se diferir, todos os blocos do cache cliente são invalidados, e re-acessados
- O Servidor também não confirma a conclusão de um write (i.e. quando o dado é alterado no arquivo local)

19

Caching

- Se um bloco de arquivo é modificado no cache do cliente, é marcado como tal e escalonado para ser copiado (flush) para o servidor
- Flush é feito em intervalos de tempo mas garantidamente antes de um close()

Entradas de diretório:

- são cachedas para leitura, mas são atualizadas diretamente no servidor
- Ao abrir um arquivo, o cache do seu diretório pai também é validado
- Períodos de verificação de coerência de cache:
  - para arquivos (3 seg), para diretórios (30 seg)

20

Transferência de blocos

- Transferência de blocos em lotes de 8K-16K
- Busca antecipada para otimizar acesso sequencial

21

Componentes do NFS

rpc.portmaster	**Roteia procedimentos remotos para os daemons**
rpc.mountd	**Monta e desmonta sistemas de arquivos**
rpc.nfsd	**Provê acesso aos arquivos locais**
rpc.statd	**Estatísticas**
rpc.rquotad	**Verifica cotas do usuário nos volumes remotos**
mount/umount	**Monta/desmonta sistema de arquivos**
/etc/exports	**Arquivos/diretórios compartilhavéis**
/var/lock/subsys/nfs	**Bloqueia execução de várias cópias do NFS**

Inicialização

- Os programas (daemons) do NFS devem ser inicializados com o boot
- O comando pmap_dump mostra o estado dos daemons RPC (Remote Procedure Calls) do sistema.
- O script nfs em /etc/rc.d/nfs pode ser usado para interromper, reiniciar, parar ou consultar os programas NFS
  - ./nfs [start | stop | status | restart | reload]

/etc/exports

- Arquivo `/etc/exports` usado pelos daemons mountd e nfsd para determinar quais arquivos poderão ser montados, e por quais clientes
- Formato do arquivo
  - Em cada linha, uma partição ou (sub)diretório exportado
  - Lista de máquinas clientes (ou domínio) autorizados
    - opções (entre parênteses) indicam qual é o tipo de acesso
  - Exemplo

```
/home     *.inf.puc-rio.br (rw),  exu.inf.puc-rio.br (ro)
/usr/local 192.168.0.1 (rw)
```

/etc/hosts.allow e /etc/hosts.deny

- Indicam quais máquinas na rede podem usar quais serviços do servidor NFS, e quais não podem
- Serviços são o portmap, mountd, rquotad, statd
- Quando há acesso por cliente, é verificado se seu endereçoIP/nome:
  - está em hosts.allow -> permite acesso
  - Está em hosts.deny -> bloqueia acesso
  - Senão -> permite acesso
- Exemplo:

```
portmap: 192.168.0.1
mountd: ALL
```

NFS: Prós & Contras

Prós:

- Aplicação no cliente não precisa saber em qual servidor os arquivos estão (Transparência de localização e de migração)
- Reduzido overhead de conexões, pois cliente consegue acessar vários arquivos pela mesma conexão
- NFS permite o acesso a partes do arquivo (ao contrário de FTP e HTTP, que acessam arquivos inteiros)

Contras:

- Servidor NFS confia “cegamente” que o cliente já verificou o controle de acesso do userID. Não há o uso de credenciais ou senhas.
- Em redes com alta latência, NFS acaba sendo inviável, por depender do pouco eficiente RPC
- Dados são transmitidos sem critografia

26