Laboratório de Redes de Computadores e Sistemas Operacionais

Linux: O Sistema de Arquivos de Rede

Fabricio Breve

Introdução

- NFS (Network File System)
  - Permite compartilhar sistemas de arquivos entre computadores
  - É quase transparente para o usuário
  - “Sem estado”: nenhuma informação é perdida quando um servidor NFS trava
    - Os clientes podem aguardam o servidor voltar e continuam a trabalhar como se nada tivesse acontecido
  - Introduzido pela Sun em 1985
    - Implementado originalmente como substituto para máquinas cliente sem disco
    - Protocolo mostrou ser bem projetado e útil como solução genérica para compartilhamento de arquivos
  - Praticamente toda distribuição Linux moderna conta com pelo menos um suporte mínimo a NFS

Informações Gerais

- NFS é formado por uma série de componentes que incluem:
  - Protocolo de montagem
  - Servidor de montagem
  - Deamons que coordenam o serviço de arquivo básico
  - Utilitários para diagnóstico
- Parte do software (tanto no cliente quanto no servidor) residem no kernel
  - Essas partes não precisam de configuração e são transparentes sob o ponto de vista do administrador

Versões do NFS

- Lançamento público original: versão 2
- No início dos anos 90 foi lançada a versão 3
  - Aumento de desempenho
  - Maior suporte para arquivos grandes (> 2GB)
- Versão 2: não pode supor que uma operação de gravação esteja completa até receber confirmação do servidor
  - Atraso significativo nas gravações NFS
- Versão 3: elimina o gargalo, torna gravações assíncronas seguras
  - Bem mais rápido que versão 2
- Versão 3 interopera com versão 2 (apenas utiliza a versão anterior)
- Linux suporta ambas as versões a partir do kernel 2.6
- O NFS 4 está em desenvolvimento e é suportado parcialmente pelo kernel 2.6 em diante

Transporte

- O NFS roda sobre o protocolo RPC (Remote Procedure Call) da Sun
  - Define uma maneira independente de sistema para os processos se comunicarem através de uma rede
  - É possível usar tanto TCP quanto UDP como protocolo de transporte
    - Originalmente o NFS usava UDP (melhor desempenho nas redes locais e computadores dos anos 80)
    - O NFS faz sua própria remontagem de seqüência de pacotes e verificação de erros, mas não tem algoritmos de controle de congestionamento (essenciais em uma grande rede IP)
    - TCP foi introduzido para solucionar essa limitação e também para permitir o uso do NFS através de roteadores na Internet
    - Os servidores que suportam TCP geralmente também aceitam UDP, portanto a decisão fica para o cliente

Bloqueio de Arquivos

- Chamadas de sistema flock e/ou lockf
- Não funcionam de maneira perfeita em sistemas locais
  - No NFS é ainda mais instável
  - Por projeto um servidor NFS não tem informação de estado
    - Não tem a mínima idéia de quais máquinas estão usando um determinado arquivo
    - Porém isso é necessário para implementar o bloquei de arquivos
    - Solução:
      - Implementar o bloquei de arquivos separado do NFS
      - A maioria dos sistemas oferecem dois deamons para isso:
        - » Lockd
        - » Statd
      - Ambos tentam ser bem-sucedidos, no entanto o bloqueio de arquivos no NFS ainda é considerado frágil

Cookies de montagem sem estado

- Um cliente tem de montar um sistema de arquivos NFS explicitamente antes de usá-lo, da mesma forma que tem que montar um sistema de arquivos local
- O servidor não controla quais clientes montaram quais sistemas de arquivos
  - O servidor apenas expõe um “cookie secreto” no final de uma montagem bem sucedida
  - O cookie identifica o diretório montado para o servidor NFS, e assim fornece uma maneira para o cliente acessar seu conteúdo
  - Demonstar e remontar um sistema de arquivos no servidor normalmente altera seu cookie
  - Usando o cookie o cliente usa o protocolo RPC para fazer solicitações de operações no sistema de arquivos
  - O servidor não se importa com que solicitações o cliente faz ou deixa de fazer, o cliente é responsável por garantir que o servidor confirme suas solicitações de gravação antes de excluir a cópia local dos dados

Convenções para atribuição de nomes para o sistema de arquivos

- É mais fácil gerenciar o NFS com um sistema de nomes padrão
- Tipicamente você poderia criar um diretório com o nome da máquina a ser montada, por exemplo:
  - /batman/tools para o sistema de arquivos que contém as ferramentas da máquina batman
    - Dessa forma um usuário poderia interpretar uma mensagem como: “Nesta sexta-feira a noite a máquina Batman estará fora do ar para manutenção” como “Não poderei usar /batman/tools/TeX na sexta para fazer meu trabalho, em vez disso vou jogar sinuca”

Segurança e NFS

- O NFS é uma maneira conveniente de acessar arquivos em uma rede
  - Portanto possui grande potencial para causar problemas de segurança
  - Foi projetado sem preocupações com segurança (preço da conveniência)
  - Linux implementa alguns recursos para reduzir fragilidades do NFS
- O acesso aos volumes NFS é dado por um arquivo `/etc/exports`
  - enumera nome de hosts (ou endereços IP) de sistemas que podem ter acesso ao sistema de arquivos de um servidor
    - Fragilidade: servidor confia nos clientes para informarem quem são
    - É fácil fazer os clientes mentirem sobre suas identidades
    - Além disso você deve ter o cuidado de exportar somente para os clientes que confia, tomando cuidado para não exportar acidentalmente para o mundo todo

Segurança e NFS

- Como acontece com sistemas de arquivos locais, o controle de acesso em nível de arquivo é controlado pelos UIDs e GIDs e as permissões do arquivo.
  - O NFS confia no cliente para informá-lo de quem está acessando os arquivos
    - Eis aqui a importância de manter UIDs e GIDs únicos em toda a rede
    - Usuários com acesso root em um sistema podem mudar seu UID para o que bem entender, e assim o servidor inocentemente dará acesso aos arquivos correspondentes
  - Se você tem um firewall, bloqueie o acesso por TCP e UDP à porta 2049, que são usadas pelo NFS, para máquinas externas

Acesso root e a conta nobody

- Como padrão o servidor Linux NFS intercepta solicitações que chegam feitas em nome de UID 0 e as altera para que pareçam vir de outro usuário
  - Essa modificação é chamada “squashing root” (encuralando o root)
    - A conta root fica limitada às habilidades de um usuário comum
    - Uma conta substituta chamada “nobody” é definida para esse pseudo-usuário
    - O UID tradicional de nobody é 65534
      - Pode ser alterado através de opções de exportação `anonuid` e `anongid`
      - Também podemos usar a opção all_squash para associar os UIDs de todos os clientes ao mesmo UID no servidor
        - » Útil para um sistema de arquivo de acesso público
      - Temos a opção no_root_squash que desabilita a associação do UID para root
        - » Perigosa, mas as vezes necessária

NFS no lado do servidor

- Servidor “exporta” um diretório quando o torna disponível para o uso de outras máquinas
- Existem dois processos totalmente distintos para
  - Montar um sistema de arquivos (tomar conhecimento do cookie secreto)
    - **rpc.mountd**
  - Acessar os arquivos
    - **rpc.nfsd**
- Ambos dependem do RPC como protocolo adjacente, e portanto necessitam que `portmap` esteja rodando

NFS no lado do servidor

- No servidor, ambos **mountd** e nfsd devem iniciar quando o sistema inicializa e permanecer em execução
- Os scripts de inicialização do sistema normalmente executarão os deamons para você, se você tiver alguma exportação configurada
- No Red Hat os scripts de inicialização são chamados:
  - **/etc/init.d/nfs**

NFS no lado do servidor

- mountd e nfsd compartilham um único banco de dados de controle de acesso que informa quais sistemas de arquivos devem ser exportados e quais clientes podem montá-los
- Essa lista normalmente fica em **/etc/exports**
  - Essa lista normalmente é exportada automaticamente na inicialização
  - Caso tenha modificado o arquivo exports ou queira executar a exportação manualmente utilize **exportfs –a**
  - Também é possível fazer a operação uma única vez especificando as opções diretamente na linha de comando de **exportfs**

NFS no lado do servidor

- Qualquer diretório pode ser exportado (não precisa ser o ponto de montagem)
- Cada partição precisa necessariamente ser exportada separadamente
  - Exemplo: Se você tem uma partição ***/home***, você poderia exportar o ***/*** sem exportar ***/home***
- Clientes normalmente tem opção de montar um subdiretório do diretório exportado caso desejem, embora o protocolo não exija esse recurso

O arquivo **exports**

- Os clientes que podem ter aceso a um dado sistema de arquivos são apresentados em uma lista separada por espaços em branco. Cada cliente é seguido por uma lista de opções separadas por vírgula e entre parênteses. As linhas podem ser continuadas com uma barra invertida

```
/home/fbreve      juquinha(rw,no_root_squash) john(rw)
/usr/share/man    *.santalucia.br(ro)
```

O arquivo **exports**

Especificações de clientes no arquivo /etc/exports.

Tipo	Sintaxe	Significado
Nome de host	*hostname*	Hosts individuais
Grupo de redes	@ *groupname*	Grupos de rede NIS; consulte a página 372 para obter maiores detalhes
Curingas	* e ?	FQDNs[a] com curingas. "*" Não coincide com um ponto.
Redes IP	*endereçoIP/máscara*	Especificações tipo CIDR (por exemplo, 128.138.92.128/25)

a. Nomes de domínios totalmente qualificados.

O arquivo **exports**

Opções comuns para exportação.

Opção	Descrição
`ro`	Exporta somente leitura
`rw`	Exporta para leitura e gravação (o padrão)
`rw=` *`lista`*	Exporta na maior parte somente para leitura. *lista* enumera os hosts que têm permissão para montar para gravação; todos os demais têm de montar somente para leitura.
`root_squash`	Associa ("squashes", ou seja, "encurrala") UID 0 e GID 0 aos valores especificados por `anonuid` e `anongid`. Este é o padrão.
`no_root_squash`	Permite acesso normal por parte de root. Perigoso.
`all_squash`	Associa todos os UIDs e GIDs a suas versões anônimas. Útil para suportar PCs e hosts monousuários não confiáveis.
`anonuid=` *xxx*	Especifica o UID para os quais roots remotos devem ser encurralados
`anongid=` *xxx*	Especifica o GID para os quais roots remotos devem ser encurralados
`secure`	Requer que o acesso remoto se origine de uma porta privilegiada
`insecure`	Permite acesso remoto de qualquer porta
`noaccess`	Impede o acesso a este diretório e seus subdiretórios (usado com exportações aninhadas)

nfsd: servir arquivos

- Assim que uma solicitação de montagem é validada por **mountd**, o cliente pode solicitar várias operações do sistema de arquivos
  - Tais solicitações são manipuladas por **nfsd**
    - Esse deamon não precisa ser executado no cliente
    - **nfsd** recebe como parâmetro um argumento numérico que especifica quanto threads deve iniciar
      - Números muito altos ou muito baixos prejudicam o desempenho do NFS
      - 8 threads é um bom número para um servidor usado como pouca freqüência
      - Em ambiente de produção, 12 a 20 é um bom número
      - Se a carga do sistema estiver aumentando muito é sinal que você foi longe demais
      - Você pode executar **nfsstat** para verificar problemas de desempenho

NFS no lado do cliente

- Sistemas de arquivos são montados quase que da mesma maneira que sistemas de arquivos locais
  - **mount** entende a notação *nome_do_host:diretório* como o caminho do *diretório* na máquina *nome_do_host*
  - Após a montagem, o sistema de arquivos NFS é acessado da mesma maneira que um sistema de arquivos local
  - Para verificar se um servidor exportou corretamente seu sistema de arquivos, utilize o comando **showmount** no cliente
    - Exemplo:

      showmount –e coyote
  - Para montar efetivamente o sistema de arquivos você usaria algo como:

    mount –o rw,hard,intr,bg coyote:/home/fbreve /coyote/home/fbreve

NFS no lado do cliente

Flags de montagem NFS.

Flag	Descrição
`rw`	Monta o sistema de arquivos para leitura-gravação (tem de ser exportado desta maneira).
`ro`	Monta o sistema de arquivos somente para leitura.
`bg`	Se a montagem falhar (o servidor não responder), continue tentando em segundo plano e continue com outras solicitações de montagem.
`hard`	Se um servidor sair do ar, as operações que tentam acessá-lo serão bloqueadas até que o servidor volte a entrar em operação.
`soft`	Se um servidor sair do ar, as operações que tentam acessá-lo falharão e retornará um erro. Este recurso é útil para evitar que processos fiquem "pendurados" em montagens não essenciais.
`intr`	Permite que os usuários interrompam operações bloqueadas (e façam com que elas retornem um erro).
`nointr`	Não permite que o usuário provoque uma interrupção.
`retrans=`*n*	Especifica o número de vezes para repetição de uma solicitação antes de retornar um erro num sistema de arquivos montado com a opção soft.
`timeo=`*n*	Configura o período de timeout (em décimos de segundo) para as solicitações.
`rsize=`*n*	Configura o tamanho do buffer de leitura para *n* bytes.
`wsize=`*n*	Configura o buffer de gravação para *n* bytes.
`nfsvers=`*n*	Seleciona as versões 2 ou 3 do protocolo NFS (normalmente de forma automática).
`tcp`	Seleciona o transporte via TCP. UDP é o padrão.[a]

a. Note que a maioria dos servidores NFS para Linux não suporta o TCP como protocolo de transporte.

NFS no lado do cliente

- Sistemas de arquivos montados com a opção hard podem fazer com que os processos fiquem pendurados quando seus servidores ficarem inativos
- Soft tem utilidade em alguns casos, porém pode ter efeitos como colaterais, como abortar um longo processo por conta de um problema transitório na rede
  - Além disso soft prejudica alguns dos objetivos originais do NFS: confiabilidade e ausência de estados

NFS no lado do cliente

- O tamanho do buffer de escrita e gravação vale tanto para TCP quanto para UDP
  - Como o TCP é mais confiável, podemos usar valores altos como 32K
  - Para o UDP, quando clientes e servidores estão na mesma rede o ideal é algo em torno de 8K
    - O padrão é 1K, mas o manual recomenda aumentar para 8K para melhorar o desempenho

Referências Bibliográficas

- NEMETH, Evi.; HEIN, Trent R.; SNYDER, Garth. *Manual Completo do Linux: Guia do Administrador*. Makron Books, 2004.