Aula 23

Sistemas Operacionais Distribuídos

Prof. Sérgio Vale Aguiar Campos
scampos@dcc.ufmg.br

SOs de Rede

Em sistemas operacionais de rede você sabe quando é local e quando é remoto.

Assim, o trabalho não muda, com exceção de comandos para acesso remoto:

- telnet
- ftp
- etc.

SOs de Rede

Em um SO distribuído vários outros serviços devem ser implementados:
- Migração de dados, processos...
- Sistema de arquivos distribuídos transparentes;
- etc.

Mais complicado; nível mais elevado

Migração de Dados

Arquivos e dados dinâmicos devem ser compartilhados:

Arquivos: são transferidos quando necessário
- Podem ser transferidos por inteiro (afs)
- ou só as partes sendo usadas (nfs)

Dados dinâmicos:
- Dados oriundos da sincronização entre processos
- Dados referentes ao problema sendo resolvido, p.ex. matrizes comuns etc.

Migração de Dados

Tanto para arquivos quanto para dados, existem problemas de:
- consistência
- cache
- desempenho

Dados também têm que ser “traduzidos” no caso de SOs distribuídos heterogêneos.

Arquitetura Cliente-Servidor

SOs distribuídos se baseam em cliente/servidor:

- Um computador requisita dados/informações/serviços
- Outro fornece esses valores
- O primeiro envia um pedido ao segundo
- Que processa o pedido e responde

Arquitetura Cliente-Servidor

Cliente/Servidor:

- Conceitos simples
- Uso muito geral: praticamente qualquer recurso pode ser compartilhado dessa forma
- Permite implementações de sistemas totalmente:
  - Distribuídos: sistema de arquivos
  - Centralizados: web.

Migração de Computação

Às vezes é melhor transferir o processamento ao invés dos dados:

- Processador local está sobrecarregado
- Informação sobre o processamento é pequena mas dados são grandes
- Método típico: **RPC – Remote Procedure Call**

Migração de Computação - RPC (Remote Procedure Call)

Processo no processador **A** quer chamar rotina no processador **B**

A envia uma mensagem a **B** com o nome da rotina e os parâmetros e bloqueia esperando resultado

B recebe a mensagem, executa a rotina e envia mensagem com resultados

A então continua execução

Migração de Processos

Extensão lógica: quando um processo é criado ele pode ser movido para outros computadores

Motivos:

- **Balanceamento de carga**: evitar que um processador só assuma todo o trabalho
- **Speedup**: programação paralela
- **Preferência de hardware**: ex. usar o CENAPAD
- **Preferência de software:** às vezes programas utilizados estão disponíveis somente remotamente
- **Acesso a dados:** outras vezes os dados são difíceis de mover, p.ex. servidores web

Migração pode ser transparente ou não.

Sistemas de Arquivos Distribuídos

Uma das aplicações mais comuns de SOs distribuídos. Motivo:

- Disquetes são horríveis!!!
- ftp não é prático: off-line, problemas de duplicação de arquivos

Novo paradigma: E que tal se todos tivessem discos remotos?

- Tempo de acesso aceitável
- Fácil manutenção
- Servidores de arquivo dedicados

Cliente/Servidor: servidor de arquivos, usuário é cliente

- Servidor pode ser dedicado ou não

Cadê meu arquivo?

E como dar nome a arquivos remotos?

O nome pode ser:
- Transparente: não diz onde o arquivo está
- Independente: não muda se o arquivo se mover

Independência é um conceito mais forte.
- Mobilidade de arquivos é raramente implementada
- Mas provê um nível mais alto de abstração

E como ele se chama mesmo?

Existem 3 maneiras mais comuns de se dar nome aos bois:

- nome do servidor faz parte do caminho;
- diretório remoto ligado a um diretório local;
- um nome global, transparente e independente.

E como ele se chama mesmo?

nome do servidor faz parte do caminho;
diretório remoto ligado a um diretório local;
um nome global, transparente e independente.

Exemplo:

- `/../b/usr/scampos`

Simples, mas não é transparente

E como ele se chama mesmo?

nome do servidor faz parte do caminho;

diretório remoto ligado a um diretório local;

um nome global, transparente e independente.

Exemplo: NFS

```
mount serv:/usr/joe /usr/joe
```

`automount` torna mais flexível

Problema: cada computador tem uma visão diferente do sistema de arquivos

E como ele se chama mesmo?

nome do servidor faz parte do caminho;
diretório remoto ligado a um diretório local;
um nome global, transparente e independente.

Exemplo:

```
/afs/cs/usr/campos
```

Nome Global - Implementação

Uma maneira de se implementar a terceira opção acima é usar um **identificador de arquivos independente da localização**:

Por exemplo, um número **único** que identifica o arquivo (como um i-node number)

Uma tabela que mapeia deste número para a localização do arquivo
- Essa tabela pode ser distribuída

Permite mudanças de arquivos através de atualização destes arquivos

Acesso Remoto

Método usado: cliente/servidor.

Mas para melhorar o desempenho, usa-se cache:

E o tamanho do bloco? Pode ser grande, p.ex. no Andrew o bloco é de 64K

E a consistência do cache? Quando as modificações são vistas pelos outros nós?
- Imediatamente: muito tráfego. NFS
- De vez em quando: como um `sync` do Unix. Sprite grava de 30 em 30 segundos
- Quando o arquivo é fechado: muda a semântica. Andrew.

Cache fica na memória ou no disco local?

Protocolos com e sem Estado

Existem duas maneiras de se implementar o servidor:

- Com estado: servidor mantém uma lista de arquivos abertos por cliente, caches dos dados
  - Controle de acesso fica como servidor
  - Exemplo: Andrew
- Sem estado: servidor **não** guarda informações sobre clientes.
  - Controle de acesso fica com o cliente
  - Exemplo: NFS

Protocolos com e sem Estado

Servidores com estado são mais eficientes.
- Caches otimizam acesso
- Mensagens são menores

Mas servidores sem estado são mais resistentes
- Ignoram queda de servidores