CEFET – Centro Federal de Educação Tecnológica de Minas Gerais
Curso: Engenharia de Computação
Professora: Anolan Y. Milanés Barrientos
Alunos: Carlos Augusto Guerra e Diego Dinarte

Implementação de um serviço de consultas a um Log distribuído

Introdução

Segundo Anolan Y. Milanés Barrientos, a depuração em Sistemas Distribuídos é maioritariamente baseada em Logs. Cada máquina cria um ou mais arquivos locais nos quais acumula mensagens de status, de erro e qualquer coisa que for relevante. Esses logs podem ser consultados remotamente. Na indústria, para garantir que o programa faz o que a gente quer se usam testes de unidade (Unit Tests). Um Unit Test é um trecho de código que chama uma pequena unidade, tipicamente um pequeno módulo, e automaticamente verifica que ele produz as saídas desejadas para entradas apropriadas. Espera-se que eles explorem a maior parte dos caminhos possíveis no código. Eles devem executar sem intervenção humana.

Problema

Escrever um programa (usando RPC) que permita consultar arquivos de log distribuídos em múltiplos computadores, desde qualquer um deles. É necessário usar usar N>5 máquinas, cada uma gerando um arquivo de log (maquina.i.log), onde i é o identificador da máquina. Após abrir um terminal em qualquer uma dessas máquinas você deve ser possível executar um comando grep (com opções suficientes para fazê-lo usável, inclusive a opção --help, e expressões regulares) que executa em todos os arquivos de log em todas as máquinas e produz a saída no terminal (com as linhas apropriadas para especificar de qual log foram extraídas).

Objetivos

Objetivo Geral:

O Objetivo geral deste trabalho consiste no projeto e implementação de um serviço de log distribuído.

Objetivos Específicos:

- Entender funcionamento do comando grep
- Modificar cliente para enviar comando grep para o servidor;
- Modificar servidor para executar comando grep;
- Criar arquivo de log no servidor;

Desenvolvimento

O desenvolvimento do projeto desenvolvido é dividido em duas partes. Uma parte desenvolvida no servidor e outra no cliente. Isto é descrito abaixo.

----------SERVIDOR----------

O servidor basicamente contém 3 arquivos importantes:

- *msg_server.c*: arquivo que cotém o código desenvolvido. É a partir dele que são criados os arquivos gerados automaticamente pelo make.
- *rpclog*: arquivo que contém o log no qual será feito o *grep.*
- *resultgrep*: arquivo resultado da execução do *grep*.

A abaixo será explicado o código desenvolvido em *msg_server.c*:

O servidor fica aguardando a solicitação do cliente e quando isto ocorre o servidor executa a função *retrieve_file_1_svc* para executar o *grep* e retornar o resultado.

1. Cabeçalho da função *retrieve_file_1_svc* :

```
readfile_res* retrieve_file_1_svc(request *req, struct svc_req *rqstp)
```

2. Criação do comando *grep:*

Os parametros do *grep* estão em req, sendo eles:

- req->option: tags de formatação da saída do comando;
- req->keyword: termo a ser buscado;
- req->logname: nome do arquivo que contém o resultado do comando.

```
/* Create grep command and save result in resulgrep file */
strcpy(command, "grep ");
strcat(command,req->option);
strcat(command," ");
strcat(command,req->keyword);
strcat(command," ");
gethostname(hostname, sizeof hostname);
strcat(hostname, req->logname);
strcat(command,hostname);
strcat(command," > resultgrep");
system(command);
```

3. Retorno do arquivo gerado pelo comando *grep*:
    a. O arquivo é aberto:

```
/* open resultgrep file */
file = fopen(name, "a+");
    if (file == NULL) {
    res.errno = errno;
    return (&res);
}
```

    b. Após aberto o arquivo, nele é escrito o nome do servidor e o arquivo é ao cliente.

```
/* write inside resultgrep file the hostname */
fprintf(file, "%s", hostname);

fseek (file, req->start, SEEK_SET);
bytes = fread(data, 1, 1024, file);

res.readfile_res_u.chunk.data = data;
res.readfile_res_u.chunk.bytes = bytes;

/* Return the result */
res.errno = 0;
fclose(file);
return (&res);
```

----------CLIENTE----------

O cliente basicamente contém 3 arquivos importantes:

- *msg_client.c*: arquivo que cotém o código desenvolvido. É a partir dele que são carregados os IPs a serem consultados e executadas as chamadas RPCs.
- *ips*: arquivo que contém a lista de IPs a serem consultados.
- *greplist*: arquivo resultado da execução das chamadas RPCs que contém os logs que continham o termo buscado.

A abaixo será explicado o código desenvolvido em *msg_client.c*:

O cliente lê os IPs a serem consultados e realiza a chamada RCP para cada IP carregado e armazena no arquivo *greplist* o que foi retornado pelos servidores.

1. Leitura dos IPs, salvos no arquivo *ips*, a serem consultados:

```
fp = fopen("ips","r");
printf("\n\n");
buffer = (char *)malloc(15 * sizeof(char));
```

2. Enquanto houver IPs no arquivo *ips* o cliente executa o processo de consulta de log nos servidores:

```
//while (!feof(fp)) {
str = fgetc(fp);
while(str != EOF) {
```

3. É realizada a chamada RCP:

```
result = retrieve_file_1(&req, clnt);
```

4. O resultado da chamada RCP é salvo em :

```
/* wirte data received from server inside a file */
write_bytes = fwrite(result->readfile_res_u.chunk.data, 1,
                     result->readfile_res_u.chunk.bytes, file);
fwrite("\n", 1, 1, file);
total_bytes += result->readfile_res_u.chunk.bytes;
if (result->readfile_res_u.chunk.bytes < MAXLEN)
break;
```

Conclusão

Após desenvolvimento deste trabalho, foi possível concluir que logs são muito importantes para analisar o funcionamento de cada máquina presente em uma rede. Além disso, analizar logs de todas as máquinas presentes em uma rede remotamente é muito util e automatiza o processo no qual um indivíduo deveria estar fisicamente presente em cada máquina da rede para analisar seus logs e seu status. Desta maneira, é possível ter uma visão geral e fazer um levantamento de quais problemas ocorrem em uma rede e quais ocorrem com mais frequencia.
O trabalho foi muito importante, pois contribuiu para se entender as necessidades de um log de de uma pesquisa remota de logs, além de praticar o conteúdo aprendido em sala de aula.

Referências Bibliográficas

- COULOURIS, George et al. Sistemas Distribuídos: Conceitos e Projeto . Bookman Editora, 2013.