Sistema de arquivos distribuído

CK0154 – Sistemas Distribuídos - Projeto Final

Javam Machado

1 Introdução

Um sistema de arquivos distribuído possui como objetivo prover ao usuário acesso remoto aos arquivos em uma rede de forma transparente. Para isso, é preciso que haja uma separação clara das funcionalidades implementadas entre os diferentes componentes. Os três principais componentes na arquitetura de um sistema de arquivos distribuído são:

- Módulo cliente: comunica-se com os serviços de diretório e de arquivo, estendendo suas operações e provendo uma interface única a nível de usuário para a aplicação.
- Serviço de diretório: realiza funções relacionadas à manutenção dos nomes de arquivos nos diretórios. Sua principal função é prover a resolução de nomes.
- Serviço de arquivo: realiza funções relacionadas à operação/manipulação no conteúdo dos arquivos. Possibilita a leitura, escrita, criação e remoção de arquivos.

2 Especificação

2.1 Objetivo

Para o projeto final da disciplina, uma versão simplificada de um sistema de arquivos distribuído deverá ser implementada. Nesse sistema, deverá haver quatro tipos de componentes ou processos: clientes, servidor proxy, *namenode* (nó de resolução de nomes) e *datanodes* (nós de armazenamento).

Um cliente deverá ser capaz de se comunicar com o sistema de arquivos destribuído, enviando requisições de operações sobre arquivos. Além disso, o cliente deve implementar uma interface que exponha os serviços oferecidos pelo sistema de arquivos distribuído.

O servidor proxy, por sua vez, será o responsável por atender às solicitações do cliente, devendo prover um conjunto de serviços (leitura, escrita e remoção de arquivos). Ao receber uma requisição, ele deverá descobrir o *datanode* em que o recurso solicitado está armazenado e requisitá-lo diretamente. Ele fará isso consultando o *namenode*, o qual manterá uma estrutura de mapeamento entre nomes de arquivos e *datanodes*.

Quando um novo arquivo é criado, uma entrada nessa estrutura de mapeamento do *namenode* será criada utilizando uma função *hash* sobre o seu nome.

Em Java, por exemplo, podemos usar o método `hashCode()` para essa finalidade. Suponha um sistema consistindo de 3 *datanodes*. Ao criar um novo arquivo, o *datanode* onde o arquivo será armazenado pode ser determinado utilizando a seguinte função: `hashCode("arquivo.txt") % 3`.

Os *datanodes* serão responsáveis por persistir os dados da aplicação, armazenando os arquivos de fato. Desse modo, responderão a consultas feitas pelo servidor proxy (após este descobrir o endereço do *datanode* através do *namenode*), o qual, por sua vez, devolverá os dados para o cliente.

Não é necessário se preocupar com controle de concorrência entre os clientes. O foco do trabalho é na **comunicação** entre os diferentes componentes de um sistema de arquivos distribuído.

Além disso, cada um dos componentes deve rodar como um processo isolado, mas comunicando-se com os outros componentes.

Para efeitos de simplificação, serão trabalhados apenas arquivos de texto (*strings*).

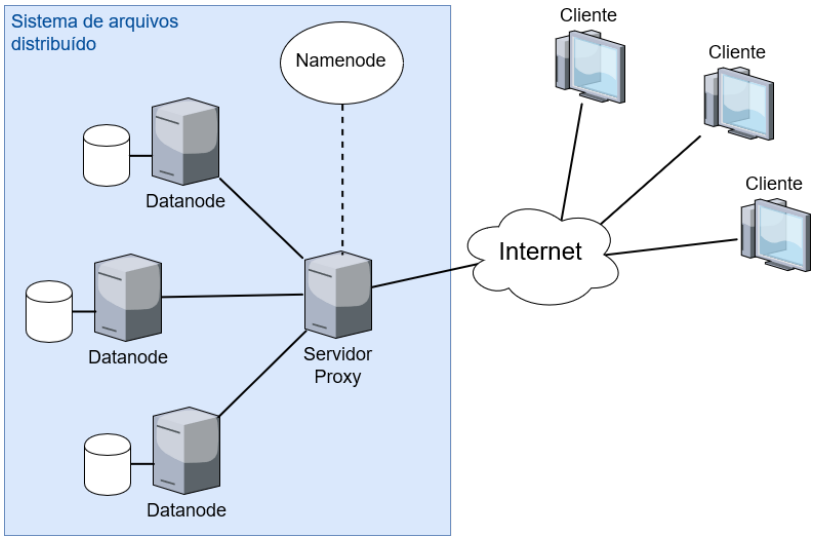

Figure 1: Exemplo de arquitetura de sistema de arquivos distribuído

2.2 Atividades

- Implementar um cliente com uma interface para solicitar as funcionalidades de escrita, leitura e remoção de arquivos;
- Implementar um servidor proxy capaz de receber e processar as requisições do cliente;
- Implementar um *namenode* capaz de realizar a resolução de nomes de arquivos para *datanodes*;
- Implementar um *datanode* capaz de armazenar arquivos;
- Realizar a comunicação entre os diferentes componentes;

- Fornecer um *log* das atividades ocorridas dentro do sistema, printando-as no *console*. Exemplo:
  - `Solicitação de leitura do arquivo "aula.txt"`
  - `Arquivo "aula.txt" encontrado no` *`datanode`* `2`

2.3 Bônus

Serão atribuídos 2,0 pontos de bônus para a correta implementação de uma estratégia de replicação. Isso pode ser feito estabelecendo, segundo algum critério, um par de *datanodes* (duas réplicas) **distintos** para cada valor de saída da função *hash*. Desse modo, todo arquivo será armazenado em pelo menos dois *datanodes*, garantindo uma certa tolerância a falhas. É importante ressaltar que a consistência entre as réplicas deve ser mantida. Logo, a remoção de um arquivo, por exemplo, deverá implicar na remoção de todas as suas réplicas.

2.4 Critério de avaliação

Para efeitos de correção, serão considerados os seguintes critérios:

- Explicação da estruturação e funcionamento do código (25%);
- Correta implementação das funcionalidades exigidas (75%):
  - Leitura (25%)
  - Escrita – criação e atualização (25%)
  - Remoção (25%)

2.5 Observações

- A comunicação deverá ser feita utilizando RPC/RMI;
- Poderão ser utilizadas as linguagens Python, C/C++ ou Java;
- O trabalho poderá ser feito em dupla ou individualmente.

2.6 Entrega

- O *upload* do trabalho para o Bitbucket deverá ser feito até o dia 4 de dezembro;
- Um repositório entitulado "SD_ProjetoFinal_[matrícula]" deverá ser criado para essa finalidade, e compartilhado com os e-mails:
  - `isabel.lima@lsbd.ufc.br`
  - `davi.braga@lsbd.ufc.br`
  - `javam.machado@lsbd.ufc.br`
- A apresentação dos trabalhos ocorrerá nos dias 5 e 7 de dezembro, no LEC, durante o horário de aula.

References

[1] George F Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed systems: concepts and design*. Pearson Education, 2005.