Universidade Federal de Santa Catarina - UFSC
Departamento de Informática e Estatística
Curso de Ciências da Computação
INE5418 - Computação Distribuída

Marina Pereira das Neves Guidolin
Weslley Silva

Trabalho 1: Backup Primário

Florianópolis, 24 de Novembro de 2020

1 INTRODUÇÃO

Esse trabalho possui o objetivo de implementar um sistema que simula a Replicação em Sistemas Distribuídos. A partir dele, foi implementada a simulação de atualizações em um arquivo replicado e as cinco fases de comunicação entre front-end e os gerenciadores de réplicas. Para isso, o grupo escolheu a linguagem Java. Além disso, o trabalho foi desenvolvido com o auxílio do framework Spring Boot.

2 FASES DE COMUNICAÇÃO

2.1 REQUISIÇÃO

Com o objetivo de simular as requisições recebidas pela aplicação distribuída, foi feita uma aplicação Web com o auxílio do Spring Boot. Essa aplicação, chamada de **primary-backup**, é a responsável pelo gerenciador de réplicas primário. A partir de requisições http enviadas pelo front-end, esse gerenciador primário é o responsável por recebê-las e tratá-las.

2.2 COORDENAÇÃO

Todas as requisições enviadas pelo front-end ao gerenciador primário possuem uma estrutura da seguinte forma:

*****verificar se pode ser assim ou se é de outra forma**

{ "id": "", "content": "" }

Ao recebê-las, é criado um identificador único para cada requisição com o objetivo de identificá-la posteriormente. O gerenciador primário trata as requisições a partir de suas ordens de chegada como o auxílio do método *addUser()*, presente na classe UserController do projeto primary-backup.

```
@RequestMapping("/user")
public User addUser(@RequestBody User user) throws IOException {
    String uuid = ""+System.currentTimeMillis();
    String content = write(user, uuid);
    sendMessage("normal", content, uuid);
    return user;
}
```

2.3 EXECUÇÃO

Após receber a requisição e criar um identificador para ela com o auxílio do método *addUser()*, o gerenciador executa uma operação de escrita em arquivo com o objetivo de simular a execução da requisição. A escrita no arquivo é realizada de forma que seja possível identificar de qual requisição partiu a operação, qual usuário que realizou e qual seu nome (“Request number: 1606220834974 UserID: 1 User's name: Marina”). Para realizar a operação de escrita, é usado o método *write()*.

```java
public String write(User user, String uuid) throws IOException {
    FileWriter writer = new FileWriter(currentDirectory + "/file.txt", true);
    String content = "UserID: " + user.getId() + " User's name: " + user.getContent();
    writer.write("Request number: " + uuid + " " + content);
    writer.write("\n");
    writer.close();
    return content;
}
```

2.4 ACORDO

A partir do momento que o gerenciador primário recebe uma requisição de atualização, o gerenciador primário envia uma mensagem às suas réplicas via multicast. Tal mensagem possui o mesmo conteúdo que foi escrito no arquivo presente no gerenciador primário. A partir disso, é possível que as réplicas tenham a informação de qual requisição foi recebida e quais as suas respectivas informações. O envio da mensagem é feito a partir do método *sendMessage()*.

```java
public void sendMessage(String message, String content, String uuid) throws IOException {
    server.createMSG("normal", content, uuid);
}
```

As réplicas de backup foram representadas por outra aplicação, que foi chamada de replicated-backup. Essa outra aplicação, rodando ao mesmo tempo que a aplicação primary-backup, recebe as mensagens enviadas pela aplicação da réplica primária, faz uma operação de escrita no seu próprio arquivo de acordo com a informação que foi recebida e envia uma confirmação de recebimento para a réplica primária a partir de uma mensagem multicast.

2.5 RESPOSTA

O gerenciador primário, ao terminar as tarefas relacionadas à request e atualizar as réplicas sobre as operações feitas, responde ao front-end retornando a informação que foi enviada e escrita no arquivo, juntamente com o código 200, mostrando que a request foi bem sucedida. Para verificar o funcionamento do retorno para o front-end, foi utilizada uma ferramenta que envia e recebe o retorno de requisições (Insomnia). Na imagem a seguir, está demonstrado como ocorre o retorno.