Relatório do trabalho prático Parte 2

Amaury Teixeira Cassola (00287704)
Arthur Bastos Fanck (00208361)
Gustavo Oliva Barnasque (00263056)

15 de Novembro de 2021

Universidade Federal do Rio Grande do Sul
INF01151 - Sistemas Operacionais II - Turma B
For: Alberto Egon Schaeffer Filho

---

1 Funções Implementadas

Esta parte do projeto, além de não alterar as funcionalidades apresentadas na primeira parte, implementa todas novas funcionalidades solicitadas. Agora, o servidor primário, além de receber conexões de clientes permite que outros servidores se conectem a ele, tornando-os assim servidores backup, diferenciando as mensagens de conexão de Servidor e Cliente pelo seu Tipo definido no Pacote enviado.

A cada nova conexão de um Servidor backup no primário, o Servidor novo é colocado em um vetor de Servidores denominado Pool e, também, suas informações (PID, Ip, Porta e File Descriptor) novo são repassadas a todos outros backups já conectados, assim, todos os backups possuirão as informações dos novo backups conectados, e o contrário não é válido. Considere a sequência de passos: Servidor 1 se conecta ao Servidor primário 0; Servidor 2 se conecta ao servidor primário 0, Servidor 1 é notificado da conexão do Servidor 2, o Servidor 2 não recebe as informações do Servidor 1. Com isso, a cada nova conexão de backup o número de servidores conhecidos pelos backups já conectados é aumentada em 1 unidade, enquanto que o backup mais recente não possuí conhecimento de nenhum outro backup. O único servidor que conhece as informações de todos backups é o Servidor Primário.

Todas mensagens enviadas por qualquer Cliente, ao servidor primário, são repassadas para os servidores backup, os mesmos realizam as operações para manter o estado consistente e retornam ao servidor principal a mensagem de sucesso. Somente após todos servidores backup realizarem o envio da mensagem de volta ao primário que o mesmo se encarrega de notificar o Cliente do sucesso, ou não, da operação.

Caso seja detectado por algum dos Backups que o Primário caiu, é realizado o processo de eleição de um novo primário. Onde é verificado, por todos os Backups, o tamanho da sua Pool (vetor de Servidores) de Backups conectados. Caso o servidor verifique que sua pool seja a de menor tamanho, com tamanho inicial sendo 0, este assume o papel de Primário realizando o fechamento do seu socket e criando um novo

socket utilizando do ip e porta do primário antigo. Caso contrário, a cada 5 tentativas de reconexão ao novo primário, o tamanho necessário da pool para o servidor se tornar o primário é aumentada em 1 unidade. Este processo se repete até que algum backup assuma o papel de primário.

Foi, também, introduzido entre o Cliente e o Servidor Primário um FrontEnd, que se encarrega de detectar a falha do Primário e armazenar as mensagens enviados pelo Cliente em uma fila, para que assim que seja reestabelecida a conexão, elas sejam enviadas. Além de armazenar as mensagens enviadas pelo Cliente enquanto o Primário está fora, tem o trabalho de tentar reconexões com o Primário a cada 2 segundos e realiza até 5 tentativas, caso nesse espaço de tempo não seja possível a reconexão, fecha a aplicação do Cliente.

1.1 Conexão de um novo Backup

O fluxo de conexão de um novo backup se dá pelos seguintes passos:

- Backup conecta via socket com o Primário;
- Primário aceita a conexão e envia dados da conexão do backup novo para todos backups já conectados;
- Primário atualiza sua Pool de backups;
- Primário retorna ao Backup o aceite da conexão.

Com isso cada backup conectado terá ciência, apenas, dos novos backups que se conectem.

1.2 Replicação passiva

Todas mensagens recebidas pelo primário, são encaminhadas aos backups, com o intuito de manter um estados consistente entre todos servidores. Somente após todos backups responderem ao primário que este notifica o cliente do aceite da mensagem. Esta funcionalidade se da pelo encaminhamento da mensagem recebida aos backups e estes a tratam de uma maneira que difere um pouco do primário. Por exemplo, no comando de envio de mensagem aos seguidores, os backups não realizam o envio das mensagens aos clientes, somente atualizam sua lista de Perfis.

1.3 Eleição

Para a eleição de um novo servidor primário, é utilizada a ideia de que o último servidor conectado assumirá o papel de primário. Como os backups somente conhecem os outros backups que se conectaram após sua própria conexão eles possuirão uma pool de tamanho inversamente proporcional ao seu número de conexão. Por exemplo, 3 backups conectados, o primeiro possuirá um pool de tamanho 2, o segundo uma pool de tamanho 1, e o terceiro uma pool de tamanho 0. Assim, o tamanho da pool é único para todos backups.

Utilizando desta propriedade criada, assim que o servidor primário cai, o backup com pool de tamanho 0 assumirá o papel de primário. Caso os backups não percebam a volta do primário em tempo determinado,

uma tentativa a cada 500 milissegundos até 5 tentativas, o tamanho considerado será 1, e assim por diante, até que algum backup assuma o papel de primário. Este algoritmo é para ser uma versão de implementação do algoritmo de bully.

Após eleito o servidor backup realizará a criação de uma nova socket utilizando dos dados do primário que havia caído. Esta nova socket possuíra ip e porta do primário que havia caído. Assim os demais servidores backup e Frontends conectados não precisarão ser notificados, pois podem utilizar das mesmas informações já existentes para conexão.

1.4 FrontEnd

Esta camada foi introduzida para o Cliente ficar alheio a indisponibilidade do servidor primário, seus papeis são: Armazenar mensagens enviadas durante a indisponibilidade e realizar a ponte entre Cliente e Servidor. As mensagens armazenadas são enviadas logo após a reconexão. No passo em que o FrontEnd detecta queda do servidor, este realiza uma espera de 2 segundos para tentar reconexão. Após primeira tentativa, a cada 2 segundos por tentativa, com o máximo de 5 tentativas, tenta realizar novas reconexões. Se, ao final deste período, não for possível reconectar ele encerra a aplicação do Cliente.

2 Estruturas Utilizadas

Além das estruturas criadas na parte 1 do trabalho foram implementadas novas estruturas.

2.1 ServerPerfil

Esta estrutura tem o efeito de armazenar as informações referentes ao um Servidor, ela possui os seguintes campos:

PID: PID do processo servidor;

Ip: IP do socket do servidor;

Port: Porta do socket do servidor;

FD: File descriptor no qual o primário utilizará para enviar mensagens a este servidor;

isAlive: Flag para verificar se o servidor ainda está respondendo ou não.

2.2 Pool (Lista de Servidores)

Todos os servidores backups conectados são mantidos em um vetor de ServerPerfil na memória do servidor. Este vetor é global e acessível por todas as threads do Servidor.

3 Ambientes de testes

Foram utilizados três ambientes de testes principais para testar o projeto ao longo do seu desenvolvimento. Foram eles:

- **Ambiente 1:** OS: Mx-Linux 19.2 v. 20200801
  Compilador: g++ version 8.3.0 (Debian 8.3.0-6)
  Thread Model do compilador: posix
- **Ambiente 2:** OS: Ubuntu 64-bit emulado através do Oracle Virtualbox
  Compilador: g++ version 8.3.0 (Ubuntu 8.3.0-6ubuntu1)
  Thread Model do compilador: posix
- **Ambiente 3:** OS: macOS Big Sur Version 11.5.1
  Compilador: clang++ version 13.0 (clang-1300.0.29.3), com as flags -std=c++11 e -stdlib=libc++
  Thread model do compilador: posix

4 Problemas Encontrados

O problema encontrado na etapa 1 do trabalho, relativo a conexão de 2 Clientes em mesmo instante a um mesmo perfil, foi solucionado com a introdução de um semáforo binário para inclusão de um Perfil novo na lista de Perfis.

Relativo a esta etapa, o principal problema foi em como notificar os clientes do novo primário eleito. Para solução deste problema adotamos que o backup eleito utilizaria do IP e porta do servidor primário que caiu para atender as conexões, assim os clientes e demais backups não precisam ser notificados onde se encontra o novo primário, dado que as informações de conexão já estão presentes em seus processos.

Outro problema encontrado foi o de eleição na parte de comunicação entre backups. Para resolver isto utilizamos da solução evidenciada no Item 1.3 deste relatório, assim comunicações entre servidores backups não foram mais necessárias. Somente precisamos da comunicação Primário -> Backup.

O item de deixar o Cliente alheio a falha do servidor foi solucionada com a introdução da fila de mensagens que o FrontEnd armazena enquanto tenta realizar a reconexão. Assim a thread que lida com o input do usuário continua captando os comandos e armazenando-os na fila, enquanto a thread de envio das mensagens se preocupa com a reconexão.