Universidade Federal
do Espírito Santo

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

Processamento Paralelo e Distribuído – Turmas 01 e 02 (EARTE) – 2021/1
Prof. Rodolfo da Silva Villaça – rodolfo.villaca@ufes.br
Trabalho T1.3 – Uso de Middleware Publish/Subscribe e Filas de Mensagens

Objetivo:
Experimentar a implementação de sistemas de comunicação indireta por meio de middleware Publish/Subscribe com Filas de Mensagens. Sincronizar a troca de mensagens entre os componentes do sistema. Utilizar brokers de troca de mensagens na implementação de sistemas distribuídos.

Instruções:

Definições Gerais
1. O trabalho deverá possuir broker local para troca de mensagens. Sugere-se usar ou um broker baseado no protocolo MQTT (Message Queuing Telemetry Transport) ou AMQP (Advanced Message Queuing Protocol). No sistema operacional Linux recomenda-se os servidores Mosquitto[1] ou RabbitMQ[2];
2. O serviço a ser implementado é uma Tabela Hash Distribuída (DHT) com espaço de endereçamento com tamanho pelo menos n=32 bits) e exatamente 8 nós virtuais (valor fixo). Lembrando que com n=32 bits de espaço de endereçamento, as chaves de hash (k) pertencem ao intervalo k = $[0..2^n-1]$ organizadas em uma topologia circular, crescente no sentido horário;
3. Os nós da DHT podem ser processos, threads, VMs ou computadores físicos (não virtuais). O sistema poderá ignorar a possibilidade de falhas nos nós da DHT: assuma que os 8 nós estarão sempre disponíveis após a etapa de inicialização da rede;
4. Cada um dos i=8 nós deverá sortear aleatoriamente um identificador de 32 bits ($NodeID_i$) no espaço de endereçamento. O NodeID serve para posicionar virtualmente o nó no espaço de endereçamento virtual da DHT. Como haverão i=8 nós, teremos o espaço de endereçamento dividido em 8 intervalos: ($NodeID_0$..$NodeID_1$], ($NodeID_1$..$NodeID_2$], ($NodeID_2$..$NodeID_3$], ($NodeID_3$..$NodeID_4$], ($NodeID_4$..$NodeID_5$], ($NodeID_5$..$NodeID_6$], ($NodeID_6$..$NodeID_7$], ($NodeID_7$..$NodeID_0$];
5. Cada nó ficará responsável por armazenar o valor v associado à chave k pertencente ao intervalo de sua responsabilidade. Lembrando que k é a chave de indexação da tabela hash e v é o valor armazenado na tabela, no formato de armazenamento chave-valor <k, v>. Considere que v é uma string (caso queira, o grupo poderá impor limite de caracteres a essa string);

1 https://mosquitto.org/
2 https://www.rabbitmq.com/

6. Os intervalos sob responsabilidade de cada nó da DHT obedecerão o seguinte relacionamento:
   - Nó 0 –> ($NodeID_7$..$NodeID_0$]; // aberto-fechado
   - Nó 1 –> ($NodeID_0$..$NodeID_1$];
   - Nó 2 –> ($NodeID_1$..$NodeID_2$];
   - Nó 3 –> ($NodeID_2$..$NodeID_3$];
   - Nó 4 –> ($NodeID_3$..$NodeID_4$];
   - Nó 5 –> ($NodeID_4$..$NodeID_5$];
   - Nó 6 –> ($NodeID_5$..$NodeID_6$];
   - Nó 7 –> ($NodeID_6$..$NodeID_7$];

7. Em resumo, se uma chave k qualquer, associada ao valor v, estiver contida no intervalo ($NodeID_{i-1}$..$NodeID_i$] então v deverá ser armazenado na tabela hash contida no nó i;

Funcionamento

1. Deverá ser implementada pelo menos (no mínimo) 1 mensagem para a inicialização da DHT: join(NodeID). Nesta mensagem, durante a inicialização da DHT, e após a geração do seu NodeID, cada nó deverá publicar no broker o seu NodeID. Além de publicar o seu próprio NodeID, todos os 8 nós deverão assinar esse tipo de mensagem para conhecer os NodeIDs dos demais nós e definir os intervalos de responsabilidade: cada nó precisa conhecer o seu antecessor e o seu sucessor no anel virtual da DHT;
2. Se o grupo julgar necessário, para fins de sincronização durante a inicialização, outras mensagens poderão ser requeridas e implementadas;
3. Os nós da DHT deverão assinar mensagens do tipo put(k,v) e get(k). A primeira é para inserir um conteúdo v na DHT por meio da chave k. A segunda é para recuperar o conteúdo v armazenado na DHT (ou nulo) a partir da chave k. Todos os nós da DHT receberão todas as mensagens de put() e get() e avaliarão se o armazenamento ou recuperação é de sua responsabilidade por meio da avaliação da chave k;
4. Os nós deverão publicar mensagens de confirmação para a mensagem put(), informando que o armazenamento foi realizado com sucesso e informando o NodeID do nó que se responsabilizou pelo armazenamento. Somente o nó responsável pelo armazenamento da chave k precisa publicar essa confirmação. Pode-se ignorar erros de armazenado (assuma que <k,v> serão sempre válidos);
5. Os nós deverão publicar mensagens de resposta para a mensagem get(), informando o valor do conteúdo v armazenado. Somente o nó responsável pelo armazenamento da chave k precisa publicar essa resposta. Pode-se ignorar erros de armazenado (assuma que <k,v> serão sempre válidos);

6. As mensagens de put(), get() na DHT serão publicadas por nós (ou processos, ou VMs, ou threads, ou computadores) externos à DHT. Estes são considerados clientes do serviço de DHT. Esses clientes deverão assinar as mensagens de resposta aos put(), get() no broker;
7. O formato das mensagens é de livre definição por cada grupo, devendo serem documentados na documentação do trabalho.

Requisitos:
1. O trabalho pode ser feito em grupos de 2 ou 3 alunos: não serão aceitos trabalhos individuais ou em grupos de mais de 3 alunos;
2. Os grupos poderão implementar os trabalhos usando qualquer uma dentre as três linguagens de programação: C, Java ou Python. Os grupos também poderão usar quaisquer algoritmos ou bibliotecas para implementar as tabelas hash locais;
3. A submissão deverá ser feita por meio de um *link* com a disponibilização do código no Github, em modo de acesso público ou, minimamente, que meu e-mail rodolfo.villaca@ufes.br tenha direito de acesso ao código;
4. A documentação para uso/teste do seu programa deverá ser feita na própria página do Github através do arquivo README[3];
5. Avaliação: Adequação aos Requisitos (30%), Legibilidade do Código (30%), Documentação (40%), Extra (10%);
6. Data de Entrega: 25/08/2021 pela Sala Virtual da disciplina no Google Classroom;

Bom trabalho!

3 https://guides.github.com/features/wikis/