MC714 - Projeto 2

João Vitor Viégas Barreira | RA: 175116

Ângelo Renato Pazin Malaguti | RA: 165429

Infraestrutura Usada:

Para a realização desse projeto, foram utilizadas 3 máquinas virtuais do tipo *e2-medium* da Google Cloud rodando Ubuntu 22.04 como sistema operacional, que são máquinas que consomem poucos recursos, portanto são baratas, e já são suficientes para executar o que é requisitado pelo projeto.

Configuração das máquinas (VMs):

Para instalar os recursos necessários para execução no Ubuntu 22.04 (LTS), foram realizados os seguintes comandos em cada uma das 3 instâncias:

```
sudo apt update
sudo apt install openmpi-bin libopenmpi-dev python3 python3-dev python3-pip
pip3 install mpi4py
```

Para cada uma das instâncias, buscando permitir a livre conexão entre elas, foi acrescentado o seguinte trecho de código no arquivo: /etc/hosts.

```
10.158.0.8 serv01
10.158.0.9 serv02
10.158.0.10 serv03
```

Foi gerado um par de chaves SRA, utilizando o comando: *ssh-keygen*. O par de chaves foi copiado e repassado a todas as outras instâncias. Posteriormente, foi criado o arquivo: *~/.ssh/authorized_keys*, e adicionado a chave pública de cada instância dentro dele. Feito esses comandos, foi realizada a primeira conexão entre as instâncias com elas mesmas, para adicionar a chave deles no arquivo *~/.ssh/known_hosts*. Após esses passos as instâncias estavam prontas para a realização da conexão.

Como executar o algoritmo:

Para executar os algoritmos, o nosso repositório do github foi clonado na pasta */home/user/* de cada máquina virtual e executados, a partir de qualquer uma das máquinas, utilizando o comando: *mpirun -n 3 --host serv01,serv02,serv03 python3 /home/user/{nome_algoritmo}.py*, onde {*nome_algoritmo}* pode ser *lamport_clock*, *mutual_exclusion*, *leader_election*.

Referências Utilizadas:

- mpi4py
    - Após explorarmos as opções entre RPC, MQTT e MPI, encontramos a biblioteca mpi4py, para Python 3, e escolhemos ela por se mostrar muito mais simples, funcional e com uma documentação melhor que as outras. Inclusive, foi possível fazer o código funcionar usando essa biblioteca com apenas 1 teste e encontramos vários exemplos de uso na internet.
- Relógio de Lamport
    - Para fazer e entender melhor o algoritmo do Relógio de Lamport usamos tanto o material visto em aula (Slides 16) como também o código da Wikipédia.
    - Usamos de base também o seguinte código encontrado no github que implementa o relógio de Lamport usando Python 3 e a biblioteca mpi4py: link do código do github. Esse código nos auxiliou a fazer o algoritmo funcionar, pois inicialmente estávamos tendo alguns erros e não estávamos conseguindo avançar.
- Exclusão mútua
    - Encontramos na mesma página da wikipédia dedicada ao Relógio de Lamport um algoritmo que utiliza exclusão mútua usando o relógio e por já termos um código com Relógio de Lamport pronto e termos entendido esse algoritmo optamos por usá-lo.
- Eleição de Líder
    - Para esse algoritmo, após lermos a página da wikipedia sobre ele, percebemos que esse problema é facilmente resolvido a partir de uma topologia em anel, onde uma mensagem informando o líder escolhido até o momento é passada adiante e, após uma volta completa, a mensagem contendo a escolha do líder definitivo circula novamente a rede e o líder assim é eleito.