Universidade Federal
do Espírito Santo

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

Sistemas Distribuídos – 2023/1
Prof. Rodolfo da Silva Villaça – rodolfo.villaca@ufes.br
Monitor: Eduardo M. Moraes Sarmento – eduardo.sarmento@ufes.br
Trabalho T2 – Aprendizado Federado Descentralizado

Objetivos:

1. Implementar um sistema distribuído, descentralizado (*serverless*) para a execução do processo de aprendizado federado;
2. Implementar algoritmos de eleição de líder/coordenador;
3. Implementar uma solução de aprendizado federado hierárquico;
4. Analisar os resultados da resolução de problemas utilizando aprendizado de máquina.

Definições Gerais:

1. O trabalho pode ser feito em grupos de até 3 alunos: não serão aceitos trabalhos em grupos de mais de 3 alunos;
2. O trabalho deve ser implementado usando a linguagem Python;
3. O problema a ser resolvido é o de classificação de imagens. Para isso usaremos o *dataset* MNIST, o mesmo utilizado no primeiro trabalho;
4. Pode-se usar a implementação do primeiro trabalho como ponto de partida para a resolução do Trabalho 2 e Laboratório VI.

Requisitos Gerais:

1. A divisão do *dataset* fica a critério de cada grupo;
2. O paradigma de aprendizado federado a ser implementado é o descentralizado, ou seja, não existe um servidor de agregação. Os clientes devem fazer a orquestração do processo de aprendizado, com a geração dos modelos parciais e do modelo global;
3. Os clientes deverão ser inicializados com: i) número de clientes a serem escolhidos em cada *round* de treinamento, o qual chamaremos de *n*; ii) quantidade mínima de clientes participando em cada *round*; iii) quantidade máxima de rounds necessários para concluir o treinamento; e iv) meta de acurácia, usada para parar o processo de treinamento;

4. A cada *round* os clientes devem eleger um participante para fazer o papel do servidor de agregação naquele *round*. O cliente eleito deve ser capaz de executar todas as funcionalidades do servidor desenvolvido no primeiro trabalho, incluindo a geração de modelos globais parciais;
5. Deverão ser inicializados pelo menos 5 clientes, executando em processos diferentes, e conectados entre si por meio de um *broker* MQTT, padrão *Publish/Subscribe* com Filas de Mensagens, usado como meio de comunicação;
6. Os clientes devem, a cada início de processo de treinamento (conjunto de rounds), apresentar-se em uma mensagem de inicialização (*InitMsg*);
7. Após os *n* clientes se apresentarem, e se tornarem conhecidos pelos demais clientes do sistema, inicia-se o processo de eleição do coordenador, por meio de mensagens de eleição, conforme adaptação da especificação do Laboratório VI – *ElectionMsg*;
8. Uma vez que o coordenador do sistema está escolhido, o cliente eleito coordenador assumirá o papel de agregador e não participará de nenhum round do processo de treinamento. Em outras palavras, uma vez eleito, o coordenador será o agregador até o fim de todos os rounds do processo de treinamento;
9. O cliente agregador deve enviar aos outros clientes o número do round atual no início de cada rodada de treinamento (*round*) e fazer a escolha, de maneira aleatória, dos clientes que irão fazer parte do conjunto de treinadores daquele *round*;
10. Os clientes escolhidos como treinadores devem treinar seus modelos usando os dados locais. Cada cliente que terminar de treinar, deve enviar os pesos do seu modelo local para o agregador;
11. O agregador deve esperar todos os clientes enviarem seus pesos e agregá-los por meio do algoritmo *Federated Average* (*FedAvg*), a definição do algoritmo se encontra no primeiro artigo da bibliografia;
12. Por último, o cliente eleito deve enviar aos outros clientes os pesos agregados;

13. Os clientes, incluindo o cliente eleito, irão atribuir os pesos agregados aos seus modelos locais, e farão a avaliação do modelo, por meio de métricas, usando seus dados locais. Os resultados atingidos após essa atualização devem ser impressos na tela de cada cliente e enviados ao coordenador;
14. Por fim, as métricas locais deverão ser agregadas em uma métrica global e comparada a meta de acurácia, concluindo, assim, um *round* do processo de aprendizado federado;

O processo descrito nos Itens 9 a 14 devem se repetir até que se atinja a quantidade máxima de *rounds* ou a meta de acurácia, sendo que esses parâmetros são atribuídos como entrada a todos os clientes em suas inicializações. Ao final do processo devem ser gerados gráficos mostrando a evolução do valor das métricas de acurácia de cada cliente durante os rounds de treinamento.

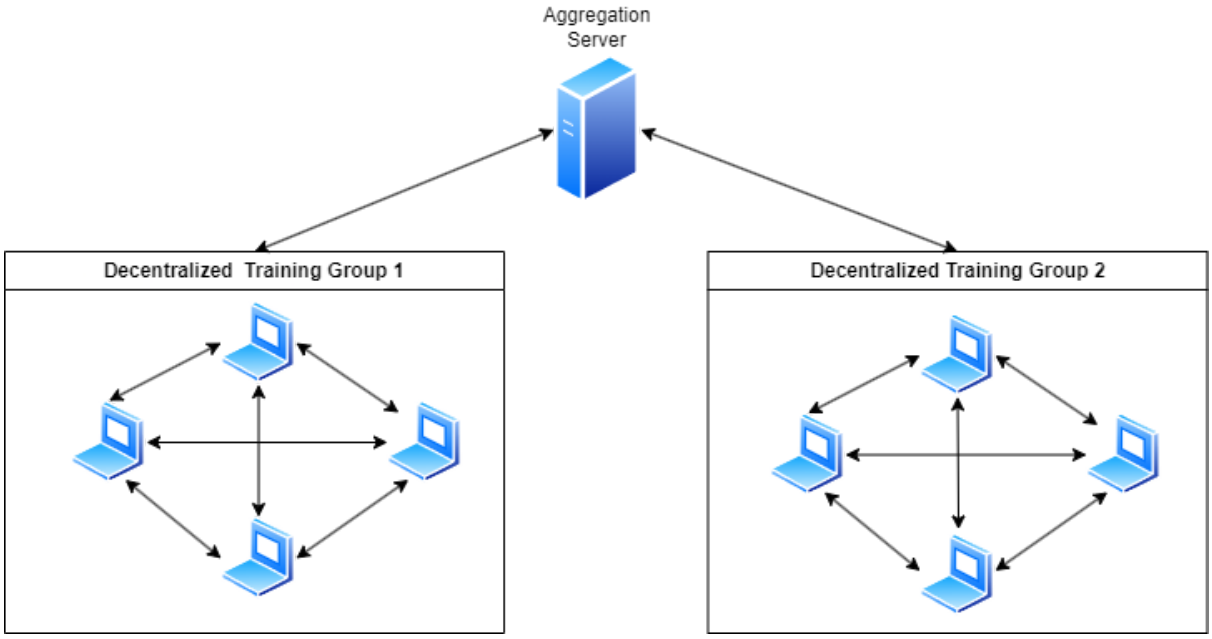
Deve-se implementar as seguintes mensagens:

1. Mensagem de Inicialização (*InitMsg*): publicada pelo cliente que está entrando no processo para os todos os outros clientes, passando o seu ID único;
2. Mensagem de Eleição (*ElectionMsg*): publicada pelos clientes contendo seu ID único e um número gerado aleatoriamente como seu voto. Cada cliente deverá esperar todos os outros clientes enviarem a mensagem de eleição e então o cliente com maior voto é eleito como cliente eleito agregador daquele round. Use a soma ID único + voto para desempates;
3. Mensagem Seleção de Treinadores (*TrainingMsg*): publicada pelo agregador para os demais clientes, passando uma lista com o ID único dos clientes escolhidos para aquele *round* de treinamento;
4. Mensagem de Fim de *Round* (*RoundMsg*): publicada pelos clientes treinadores ao agregador, enviando os pesos do modelo local e a quantidade de amostras da base de dados local usadas no treinamento;

5. Mensagem de Agregação (*AggregationMsg*): publicada pelo agregador com os pesos agregados para todos os clientes registrados, mesmo para aqueles que não foram escolhidos para o processo de treinamento;
6. Mensagem de Avaliação (*EvaluationMsg*): publicada por todos os clientes, mesmo aqueles que não foram escolhidos para o processo de treinamento, para o cliente eleito agregador passando os resultados das métricas encontradas.
7. Mensagem de Encerramento (*FinishMsg*): Caso a meta de acurácia tenha sido atingida, o cliente eleito agregador deve publicar uma mensagem para os clientes treinadores indicando a parada do processo de treinamento.

Requisitos Específicos para Pós-Graduação:

- Deve-se implementar aprendizado federado descentralizado e hierárquico. Deve-se instanciar dois grupos de treinamento distintos, isto é, um grupo de 4 clientes conectados entre si, e outro grupo de 4 clientes conectados entre si, ambos executando o aprendizado federado, mas sem qualquer comunicação direta entre eles. Um servidor de agregação central deve ser instanciado em seu próprio processo, fixo (não precisa ser eleito) e conhecido por todos os clientes de ambos os grupos de treinamento.
- A cada processo de treinamento, o cliente eleito agregador deve agregar os pesos de seu grupo e, em seguida, enviar os pesos para o servidor de agregação central. Este servidor irá agregar os pesos encontrados por todos os grupos, usando o FedAVG, e irá mandar de volta os pesos agregados para o agregador.
- O cliente eleito (agregador) em cada grupo envia os pesos vindos do servidor para todos os outros clientes do seu grupo. Todos os clientes do grupo, incluindo o eleito (agregador), atualizam seus pesos usando os pesos do servidor. Esta arquitetura pode ser vista na figura a seguir:



- Não é necessário que a comunicação entre os grupos de treinamento e o servidor de agregação seja feita usando Publish/ Subscribe;
- Para implementação da hierarquia, é necessário a implementação da seguinte mensagem:

    1. Mensagem de envio dos pesos agregados de cada grupo de treinamento enviada pelo cliente eleito ao servidor de agregação enviando os pesos do modelo global do grupo de treinamento e o somatório da quantidade de amostras das bases de dados dos clientes que participaram do treinamento. O servidor deve retornar mensagem com os pesos agregados para os clientes agregadores de cada grupo.

Entrega:

1. Por meio da Sala de Aula Virtual da disciplina no *Google Classroom*, na atividade correspondente ao Trabalho T2. 1 (uma) submissão por grupo é suficiente;

2. Deve-se submeter apenas o *link* para o repositório virtual da atividade (Github, Bitbucket, ou similares) contendo: i) códigos-fonte; ii) instruções para compilação e execução; iii) relatório técnico (*.pdf* ou *markdown, README.md*); e iv) vídeo curto (máximo 3 min) mostrando uma execução de exemplo, resultado e análise da execução;
3. O relatório técnico deverá conter: a metodologia de implementação e testes usada, resultados apresentados sob a forma gráfica, e análise e avaliação dos resultados (Ex: o resultado esperado foi alcançado? Comente!);
4. Avaliação: Adequação aos Requisitos (30%), Legibilidade do Código (30%), Documentação (40%);
5. Data de Entrega: 11/07/2023 (terça-feira);

Bom trabalho!

Bibliografia:
[1] Definição formal de Federated learning e Federated averaging:
Communication-Efficient Learning of Deep Networks from Decentralized Data
[2] Python grpc:
https://grpc.io/docs/languages/python/basics/
[3] Broker de mensagens emqx:
https://www.emqx.io/docs/en/v3.0/
[4] MNIST Dataset:
https://www.kaggle.com/datasets/oddrationale/mnist-in-csv