

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

DIM0124 - PROGRAMAÇÃO CONCORRENTE

Relatório de Desenvolvimento de Aplicação Concorrente

KNN – Nearest Neighboor

Matrícula: 20180063677

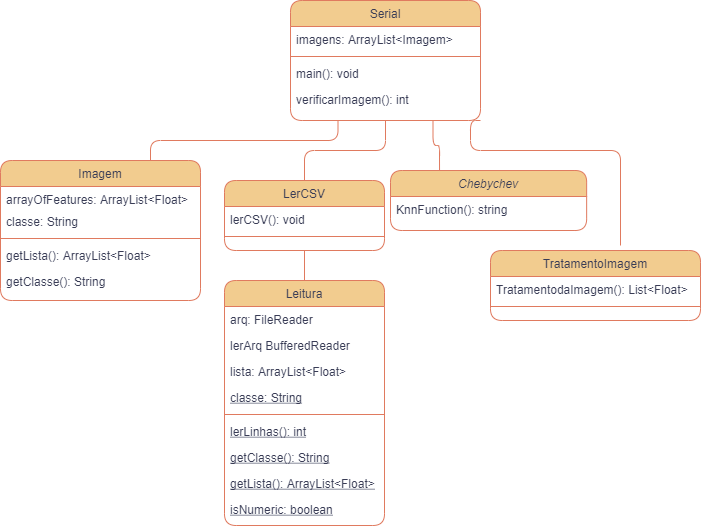
Nome: Gabriel Martins Spínola

1. **Introdução**

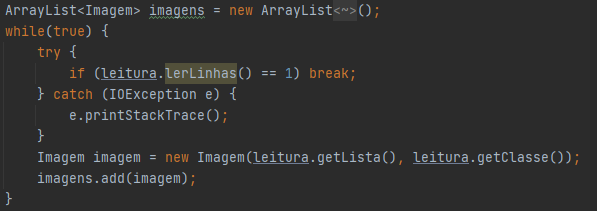
O algoritmo implementado foi o algoritmo do KNN, que tem o intuito de checar se existem ou não pessoas nas fotos que serão analisadas, para isso temos um arquivo em CSV contendo metadados de fotos com e sem pessoas para que possamos achar as fotos mais próximas quando processarmos. Como resultado obtido, o programa imprime na tela se há ou não pessoas na foto a cada imagem que for processada.

Desta vez, o algoritmo foi implementado utilizando a integração do Spark, ferramenta da Apache que permite que um programa possa ser processado em várias máquinas, que estão conectadas ao servidor Spark, ao mesmo tempo permitindo paralelizar a aplicação em nível de máquinas.

1. **Implementação Serial** 
   1. Descrição

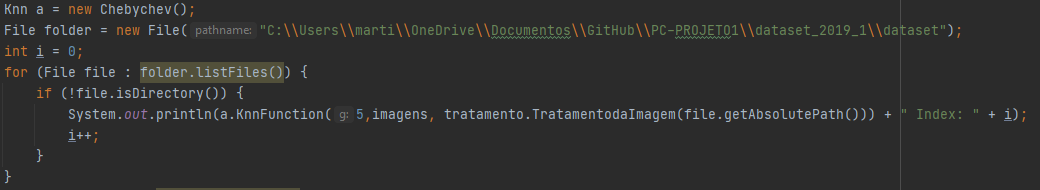


O algoritmo começa lendo um dataset em csv que possui vários metadados de algumas imagens que tenha ou não pessoas. Esse dataset é usado para calcular os vizinhos mais próximos da imagem que será processada posteriormente. Cada linha do csv é transformado em um objeto do tipo Imagem e armazenado em uma lista.

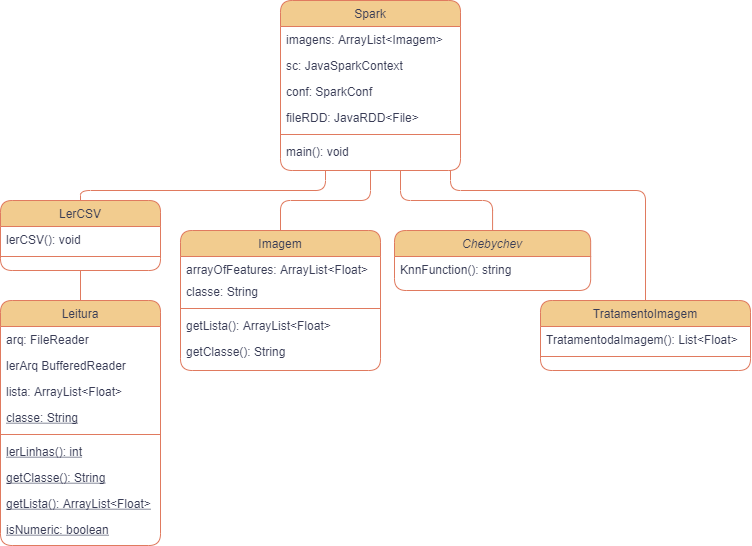


Após isso, criamos um objeto do tipo Chebchev (classe que tem o método que executa o cálculo de quais imagens são mais próximas) e percorremos a pasta do dataset executando a função de calculo para

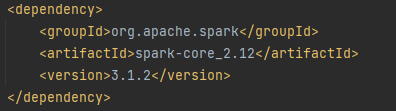
cada imagem presente no dataset e imprimindo o resultado.

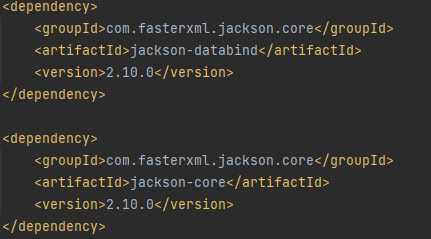


1. **Descrição da Implementação com o Apache Spark**
   1. Descrição

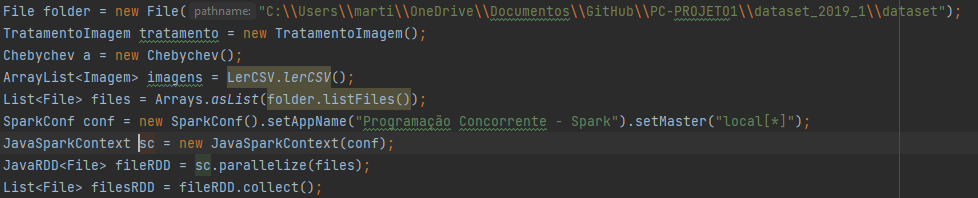


A implementação com o Spark não cria nenhuma classe a mais que a implementação serial, porém é preciso adicionar as dependências do Spark no pom.xml para ser importado automaticamente com o Maven.



Se fez necessário importar mais 2 dependências para conseguir remover alguns problemas que ocorreram quando o algoritmo era executado, as seguintes dependências:

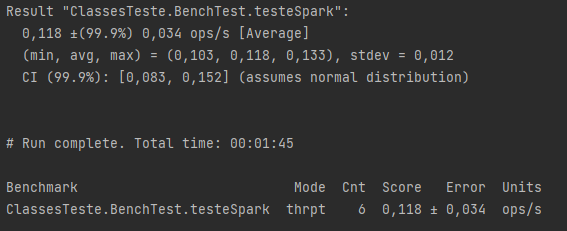
As modificações feitas em relação a implementação serial podem ser vistas a seguir:



Onde é criado um objeto do tipo SparkConf onde podemos configurar o nome da aplicação, que no caso se chama “Programação Concorrente – Spark”, e também a quantidade de núcleos que o programa irá utilizar, que no caso utilizo “local[\*]” para utilizar o máximo de núcleos disponíveis. Após coletar os dados processados apenas realizo a impressão dos resultados obtidos.

* 1. Avaliação com Microbenchmark

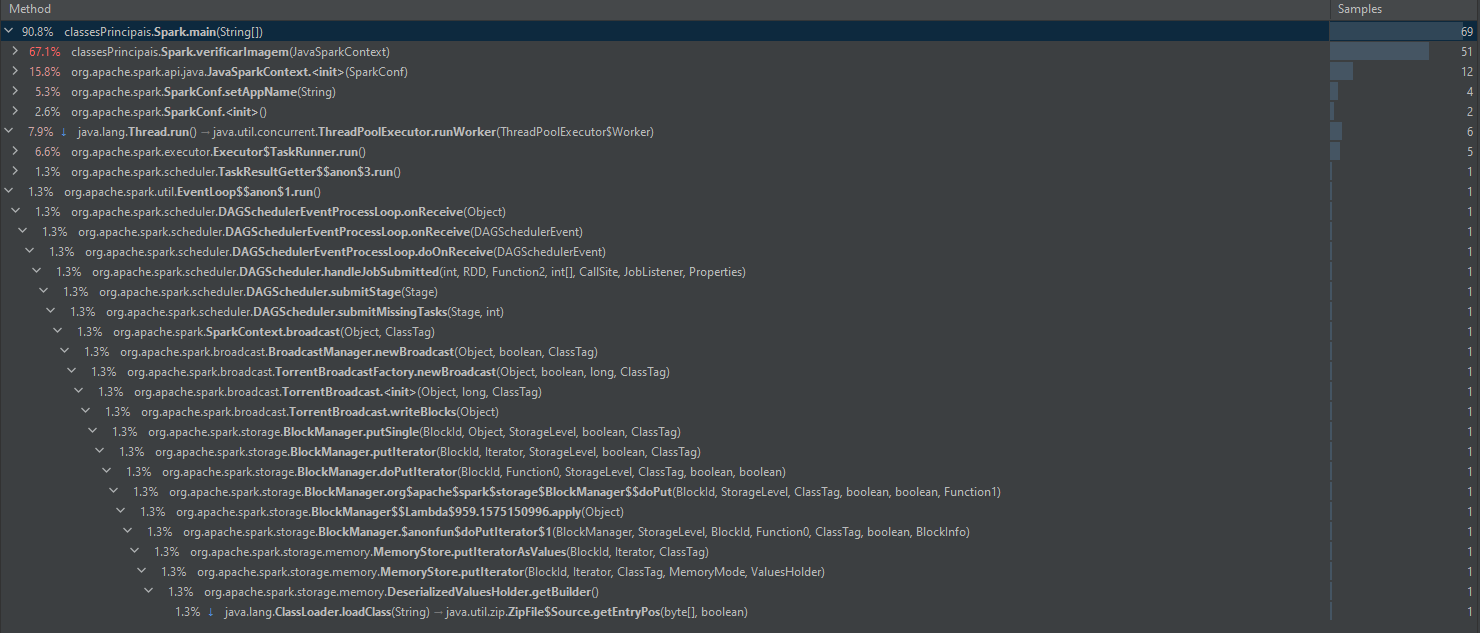
A seguir podemos ver o resultado obtido ao realizar o teste com a ferramenta do JMH no modo de throuput:



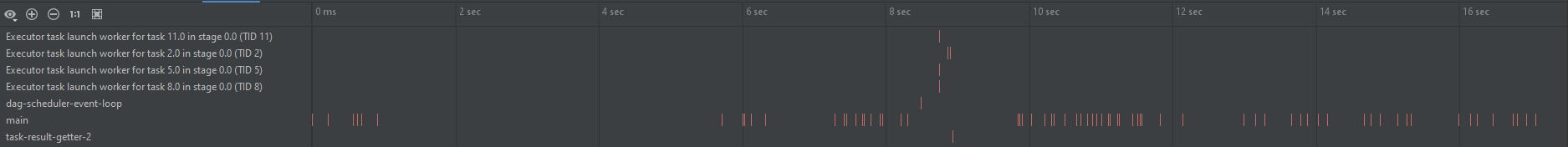
Vemos que a implementação com o Spark teve 0,118 operações por segundo com 99% de confiança com variação entre 0,083 e 0,152 vezes por segundos. Além disso apresentou taxa de erro de 0,034 operações por segundo.

* 1. Avaliação com Java Flight Recorder

Ao realizar o teste com o JFR temos as seguintes informações obtidas:



Onde vemos que a função mais chamada na execução da classe principal é a função de verificarImagem que é a função que executa toda a aplicação.



Acima podemos ver a timeline de execução do programa.

1. **Discussão**

O trabalho desenvolvido teve o intuito de praticar e aprender a programar de forma concorrente, permitindo que os algoritmos desenvolvidos fossem feitos e testados pelas ferramentas vistas em aula. Para realizar o tratamento das imagens do dataset utilizei a biblioteca openCV.

Analisando os resultados oferecidos pelo JMH pude perceber que a implementação com o Apache Spark ofereceu uma melhora no throuput, porém ao executar as versões serial e a versão com o Spark não se notou grande melhora no tempo de execução, na verdade a execução ficou um pouco mais lenta, mesmo utilizando o Spark todos os 8 núcleos que meu computador tem.

Como lição não só desse projeto final, mas da disciplina completa em si, aprendi que o ganho de eficiência pode ser significativo em uma implementação concorrente, mas irá depender muito do hardware da sua máquina. Além disso, a forma como o programador utiliza as ferramentas que as linguagens disponibilizam é essencial para se obter o melhor resultado possível, pois, ao longo da disciplina pode ser observado que algumas estratégias foram muito mais eficientes que outras para realizar os mesmos problemas. O programador também precisa estar muito atento pois muitas vezes o algoritmo pode estar sendo interrompido muitas vezes por causa de gerenciamento de memória, onde a linguagem estará rodando a coleta de lixo. Portanto, para ser um bom programador é necessário saber muito mais do que apresentar uma solução para algum problema, além de resolver o problema é preciso saber que existem várias formas de solução que pode permitir o melhor desempenhos diferentes, cabe ao programador conseguir identificar e aplicar essa forma.