Project Background

This project aims to reproduce the results of SparkBWA and StreamingBWA.

In the SparkBWA paper, the authors proposed implementing a distributed version of the BWA algorithm using Apache Spark. Compared with contemporary Hadoop- and MapReduce-based approaches such as SEAL and Halvade, SparkBWA achieves significantly faster performance by caching intermediate data in memory. In terms of implementation, the authors describe using JNI (Java Native Interface) in Scala to directly invoke the native BWA algorithm. This design makes SparkBWA independent of any specific BWA version and allows it to run without modifying BWA's source code.

In the sections on system design and performance evaluation, the authors analyze the performance of each Spark operator involved in executing BWA, including reading input files from HDFS and creating RDDs, performing join and sortByKey operations, and writing the final results back to HDFS. The experiments focus on how execution efficiency changes when allocating different computational resources to these operators. They also compare the performance difference between multi-threaded and single-threaded BWA execution when called via JNI.

In contrast, the StreamingBWA paper points out that prior distributed BWA frameworks (including SparkBWA) did not account for the time required to upload raw data to HDFS, to fetch data from HDFS during Spark execution, or to merge SAM output files. StreamingBWA addresses these bottlenecks through a streaming-based architecture.

Project motivation

Both SparkBWA and StreamingBWA were originally designed to run on distributed clusters. In particular, SparkBWA demonstrates improved performance as the number of cluster nodes increases. Given that this project will be conducted on a single laptop, with limited computational resources, we will focus primarily on operator-level performance analysis of SparkBWA. Specifically, we aim to reproduce the results presented in Section 5.2 of the SparkBWA paper, examining the join, sortByKey, and SortHDFS operators.

We will also use pBWA as a baseline for comparison, to evaluate whether SparkBWA achieves equal or better performance than pBWA under single-node conditions.

Additionally, we will attempt to reproduce part of the StreamingBWA results. Using the same laptop, we will collect and compare metrics from the Spark UI to measure throughput when processing datasets of the same size. The goal is to determine whether StreamingBWA achieves a similar 2.5× performance improvement as reported in the paper.

Reproduction and Challenges

In the experiments, we will use the ERR022066_1.filt and ERR022066_2.filt datasets from the 1000 Genomes Project, as used in the StreamingBWA paper. Our local setup consists of a Linux Mint system running on a 40 GB RAM laptop.

Before running experiments, we must first set up the Hadoop and Spark environments. It is worth noting that SparkBWA and StreamingBWA were implemented using different versions of Spark and Hadoop, so it is difficult to determine how much of StreamingBWA's performance improvement comes from the newer Spark version versus the streaming architecture itself. Nevertheless, we will use metrics exposed through Spark's monitoring UI to perform comparative analysis between the two frameworks.