Intro

Today I'm going to discuss my research on efficiently encoding a common file format used to store DNA. Let's get started.

Background

There are many methods for storing DNA on a computer. One such contemporary method is called nanopore sequencing. This technology records disturbances in ionic current as DNA molecules pass through a tiny pore.

The signal data is collected in fragments called reads, with each read stored in their own separate file. This file is in a format known as FAST5, which is actually just an HDF5 file with its own specific schema. Now, HDF5 is a scientific file format which is capable of storing anything you like, and stands for Hierarchical Data Format 5.

The problem is that the official library used to read HDF5 files is not scalable with threads. For example, here we can see that the library serialises requests from different CPUs to the scheduler. This means that any bioinformatics task using FAST5 files is slow and inefficient.

The library is also so complex with over a million lines of code that attempting to implement multithreading would be a nightmare. The solution is a simpler and much faster file format called SLOW5.

SLOW5 is a TSV file format with each read stored as an entry. Here we can see an example.

It can be read in parallel by obtaining the location and size of each read beforehand. This information is typically stored in an accompanying file known as a SLOw5 index.

However, SLOW5 is in its current readable format takes up too much disk space.

Aims

Hence, the aim of my research was to efficiently encode the SLOW5 format into a size smaller than or equal to its equivalent FAST5 files, whilst maintaining parallel access.

Method

I developed two efficient encodings of the SLOW5 format: a binary and a compressed binary version.

The binary version works as follows. It stores each numerical field in its equivalent binary representation rather than in the ASCII format. In 1 byte of data, the ASCII format can store the digits 0 through 9, whilst binary can store the numbers 0 to 255. This is one reason for the format's efficiency.

The binary encoding also removes each tab character of the TSV, since the size of most fields is now known. In the case of variable length fields, their size in bytes is prepended before the actual data.

The second encoding developed is a gzip-compatible version of the binary encoding. gzip is a well known GNU software application and file format used for lossless compression and decompression. It's based on the DEFLATE algorithm which is a combination of LZ77 and Huffman coding.

The compressed version works as follows. Each read in the binary encoded SLOW5 file is located using its SLOW5 index and then compressed separately, in order to maintain efficient parallel access.

The default level of compression, internal memory allocation and algorithm of the zlib C library were used, but different levels will be experimented with in the future.

Multithreaded access to the SLOW5 file format was then implemented using the C standard library.

The algorithm works as follows. It takes as input a SLOW5 file and a list of read IDs, and returns the corresponding reads.

To start with, a hash table of the SLOW5 index is loaded into memory. This is used to randomly access each read. Then, the list of read IDs are processed one batch at a time to avoid running out of memory. Each thread is allocated an equal amount of read IDs for the current batch. And using the hash table, each read is found and stored by the threads in parallel. Work stealing occurs between threads if one finishes earlier than another. And then finally, the stored reads are returned and a new batch begins to be processed until all read IDs have been consumed.

Results

A sample of human DNA from a typical male was used for these experiments.

The file size of these two SLOW5 encodings were compared with the original SLOW5 file and the corresponding FAST5 files.

And a list of read IDs were generated from a sequencing pipeline.

Benchmarking: - Determine access time and file size

Discussion

Conclusion