CS177H Bioinformatics: Course Project
Bowtie Implementation Using Python

Professor Jie Zheng , SIST

Xinliang Wu 42288815
Undergraduate SIST

Part(a) Introduction

This is a project report file attached to CS177H: Bioinformatics, ShanghaiTech University.

Bowtie is an ultrafast, memory-efficient short read aligner. Bowtie indexes the genome with a Burrows-Wheeler index to keep its memory footprint small.

This re-implemented version provides the function that are fundamental to Bowtie and Bowtie 2 along with some creative features which are shown as the list below

1. Index using optimized Burrows-Wheeler index to reduce memory consumption.
2. Exact Match using gained index data.
3. Finish notification supported by Python url-request module and Notification framework provided by Apple.
4. Export to Excel feature realized by xlsxwriter module.

Moreover , due to both technical and time limitations, partial functions provided by original version of Bowtie and Bowtie2 are **not** supported in the reimplemented version, which are shown as below.

1. Export the standard index file to path (the index information is stored only in memory when the py file is running)
2. Mismatch and Gap detection.
3. Any letter rather than A,T,C and G appeared in the input sequences.

For further usage document and limitations , please refer to the README file provided.

The whole project working period is about 20-days long containing 14-days of coding and test period with 7 versions of code.For detailed developing process, please refer to the git history provided as an attached file.

Part(b) Indexing Optimization

According to the original method provided in the published paper, if you want to set up an index for a text file, the first thing to do is to generate the full-rearrangement text of the original, which is always regarded as **text rotating**, however ,this method can be incredibly memory consuming for comparable larger input.

Take the input sequences with size of 10MB as an example, to store the full rearrangement text, we need to memory of:

$$MEM = (10MB)^2 = 100GB$$

This amount of memory consumption is not acceptable for such a alignment software such as Bowtie, and it makes the following sort step far more time-consuming.

Different from the original version of Bowtie and Bowtie2 , in this project , the index processing is done with some mathematical tricks instead of using the algorithm called IS in Bowtie.

Here is the detailed explanation of how the tricks work to reduce the memory consumption.

For input with size of 10MB, our final mission is to sort all the rotated text literally in order to get the Suffix Array, so it requires all the information which are sufficient to sort all the text, that is the key to reduce the memory consumption.In fact , we do not need to store the full rotated text, instead, the first X letters can provide sufficient information.

For instance, when X = 20, which means we only store the first 20 letters of a rotated file, and this provides the sufficient information to sort a text with size of :(suppose there are only ATCG appears in the text)

$$SIZE = 4^{20} = 1.099 * 10^{12} Byte \approx 1GB$$

With the memory consumption of

$$MEM = inputsize * 20$$

For an input file with size of 10MB, it reduces the memory consumption from 100GB to 200MB, which is 500,000 times smaller.

In the re-implement version, the default X is set to 20 for better balance of speed and accuracy.

Part(c) Finish Notification

For most bioinformaticians, it would be annoying for them to wait right against computer waiting the alignment to be finished. Thus Finish notification works for whom want to be instantly notified once the alignment is finished.

For more instruction on how to set up the Finish notification, please refer to the README file, this part will focus more on how the structure works.

For instance, on an iOS devices , a notification can only be pushed from Apple's notification server, a developer can apply for the certificate to connect with Apple and send notification to it. However, it would be challenging if you want to apply for a notification certificate on both time and money, thus a developer called *feng huang* contributes his own certificate with an APP called BARK.In fact, he provides a layer between Apple and user, which allows users to push notification without applying the official certificate.

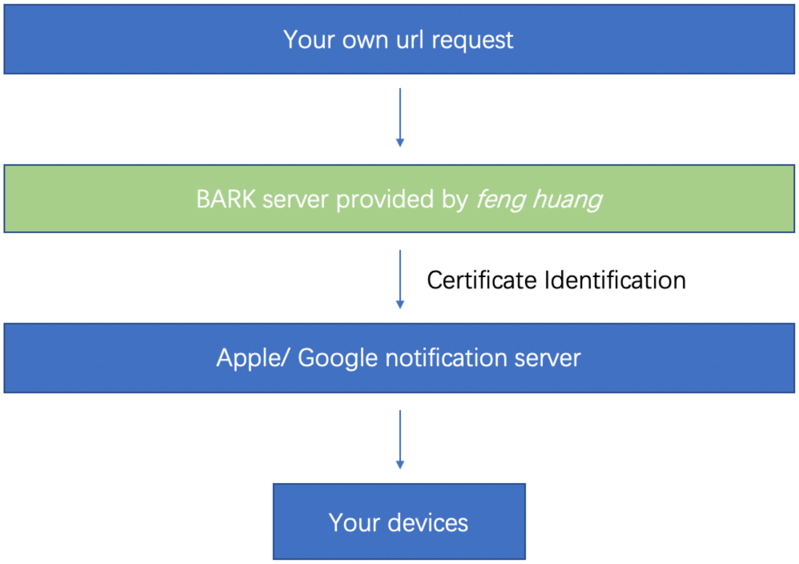

Your own url request
BARK server provided by *feng huang*
Certificate Identification
Apple/ Google notification server
Your devices

Figure 1: Notification Structure

For users using Android , due to the unstable connection with Google FCM services in Mainland China, it seems hard to build notification structure. Instead, some domestic server such as Xiaomi push and Huawei push is not free of charge, thus you would need an iOS device to use this function.

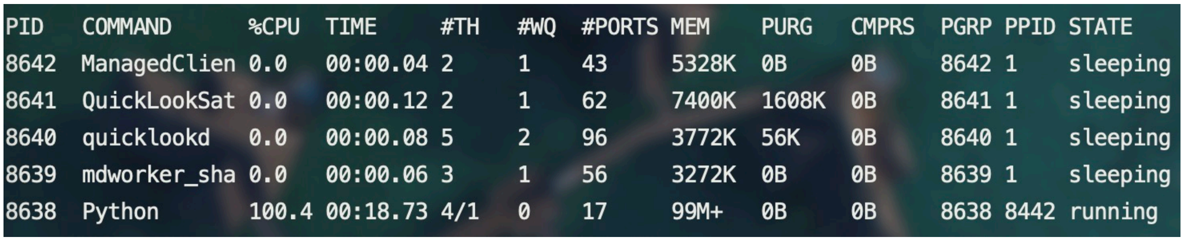
Part(d) Performance Analysis

Here is the analysis of performance based on the following perspective.

- Memory consumption
- Speed
- Accuracy

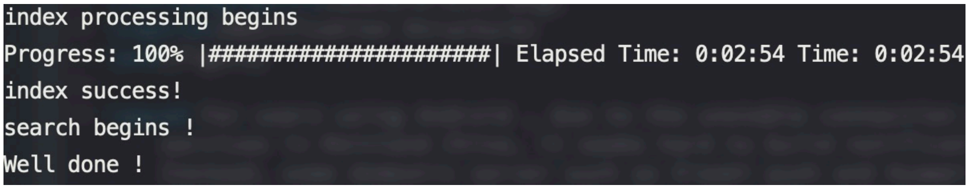
For an input file less than 500KB there are no apparent difference between the original C++ version and the python version. Here we choose 3 different kinds of input file to test how the program performed.

Case : Input reference file : 1,013,353 bp. Input compare length 195 bp
Mem: 100M — Time : 174s — Accuracy :100%



```
PID   COMMAND      %CPU  TIME     #TH   #WQ  #PORTS MEM    PURG   CMPRS  PGRP PPID STATE
8642  ManagedClien 0.0   00:00.04 2     1    43     5328K  0B     0B     8642 1    sleeping
8641  QuickLookSat 0.0   00:00.12 2     1    62     7400K  1608K  0B     8641 1    sleeping
8640  quicklookd   0.0   00:00.08 5     2    96     3772K  56K    0B     8640 1    sleeping
8639  mdworker_sha 0.0   00:00.06 3     1    56     3272K  0B     0B     8639 1    sleeping
8638  Python       100.4 00:18.73 4/1   0    17     99M+   0B     0B     8638 8442 running
```

Figure 2: Mem Consumption



```
index processing begins
Progress: 100% |#####################| Elapsed Time: 0:02:54 Time: 0:02:54
index success!
search begins !
Well done !
```

Figure 3: Speed Analysis

Note: Performance may differ on different computers, the test environment is : Intel i7-7700HQ with 16GB Ram, running MacOS 10.14.5 , python 3.7.3 installed.

Part(e) Reference

Langmead, B. (2010), Aligning Short Sequencing Reads with Bowtie. Curr. Protoc. Bioinform., 32: 11.7.1-11.7.14. doi:10.1002/0471250953.bi1107s32

Ben Langmead, Steven L. Salzberg (2012). Fast gapped-read alignment with Bowtie 2. Nature Methods, 9, 357-359.