fastqsplitter Documentation

Release 1.1.0

Leiden University Medical Center

Aug 01, 2019

Contents

Table of contents

CHAPTER 1

Introduction

A simple application to split FASTQ files.

The algorithm is a reimplementation from biopet-fastqsplitter. Fastqsplitter reads a fastq file. It then splits the reads over the designated output files.

This application does NOT work with multiline fastq sequences.

fastqsplitter uses the excellent xopen library by @marcelm. This determines by extension whether the file is compressed and allows for very fast compression and decompression of gzip files.

CHAPTER 2

Usage

```
usage: fastqsplitter [-h] -i INPUT -o OUTPUT [-c COMPRESSION_LEVEL]
                     [-t THREADS_PER_FILE]
```

2.1 Named Arguments

-i, --input The fastq file to be scattered.

-o, --output Scatter over these output files. Multiple -o flags can be used. The extensions determine which compression algorithm will be used. '.gz' for gzip, '.bz2' for bzip2, '.xz' for xz. Other extensions will use no compression.

-c, --compression-level Only applicable when output files have a '.gz' extension. Default=1

Default: 1

-t, --threads-per-file Set the number of compression threads per output file. NOTE: more threads are only useful when using a compression level > 1. Default=1

Default: 1

Note: Fastqsplitter uses a separate process for reading the input file, doing the splitting as well as one seperate process per output file. Fastqsplitter therefore always uses multiple CPU cores.

CHAPTER 3

Example

With an input file `input_fastq.gz` of 2.3 GB. `fastqsplitter -i input_fastq.gz -o split.1.fq.gz -o split.2.fq.gz -o split.3.fq.gz`

Fastqsplitter will read `input_fastq.gz`. The first 100 reads will go to `split.1.fq.gz`, read 101-200 will go to `split.2.fq.gz`, read 201-300 will go to `split.3.fq.gz`, read 301-400 will go to `split.1.fq.gz`, etc.

This way the fastq reads are evenly distributed, with a difference of maximum 100 reads between output files, and no positional bias in each output file.

CHAPTER 4

Performance comparisons

Comparing different modes of fastqsplitter and biopet-fastqsplitter. Biopet-fastqsplitter has only one mode: compression level 5, and an unknown number of threads per file.

Fastqsplitter runs with 1 thread per output file and compression level 1 by default. For fair comparison with biopet-fastqsplitter, fastqsplitter was run with 4 threads per file (xopen default) and compression level 5. Since fastqsplitter starts several pigz and one gzip process the memory usage of these processes are included in the results.

This test case was run with a 2.3 GB input fastq file zipped. This was split over 5 output files.

The used test machine had 32 GB memory (2x16GB 2133mhz), an Intel core i7-6700 (4 cores, 8 threads) and a Sandisk X400 500gb SSD.

measurement	fastqsplitter (defaults)	fastqsplitter -t 4 -c 5	biopet-fastqsplitter
real time	0m50.932s	1m28.153s	1m41.385s
total cpu time	3m7.116s	7m55.436s	8m20.304s
max mem	24 MB	32MB	400MB
max vmem	110 MB	1.6 GB	11.0 GB
output files total size	2290 MB	2025 MB	2025 MB

The outcomes for multiple runs were fairly consistent with a max +-3 second difference between runs.

CHAPTER 5

Changelog

5.1 1.1.0

- Enable the building of wheels for the project now that Cython extensions are used. Thanks to @marcelm for providing a working build script on https://github.com/marcelm/dnaio.
- Cythonize the splitting algorithm. This reduces the overhead of the application up to 50% over the fastest native python implementation. Overhead is all the allocated cpu time that is not system time.

  This means splitting of uncompressed fastqs will be noticably faster (30% faster was achieved during testing). When splitting compressed fastq files into compressed split fastq files this change will not be much faster since all the gzip process will be run in a separate thread. Still when splitting a 2.3 gb gzipped fastq file into 3 gzipped split fastq files the speedup from the fastest python implementation was 14% in total cpu seconds. (Due to the multithreaded nature of the application wall clock time was reduced by only 3%).

5.2 1.0.0

- Added documentation for fastqsplitter and set up readthedocs page.
- Added tests for fastqsplitter.
- Upstream contributions to xopen have improved fastqsplitter speed.
- Initiated fastqsplitter.