Counting Kmers for Biological Sequences at Large Scale

Abstract

A fundamental move in bioinformatics is counting the abundance of all distinct kmers in biological sequence records. SWAPCounter is a distributed solution for kmer counting that is highly scalable. When processing 4 TB of sequence data from 1000 Genomes, SWAPcounter scales to 32,768 cores with 79 percent parallel efficiency. It has competitive performance with two other tools in a shared memory environment, KMC2, and MSPKmerCounter. SWAPCounter's source code is freely accessible at https://github.com/mengjintao/SWAPCounter .

Keywords: Kmer counting , Biological sequence , Counting bloom filter , Scalability

1 Introduction

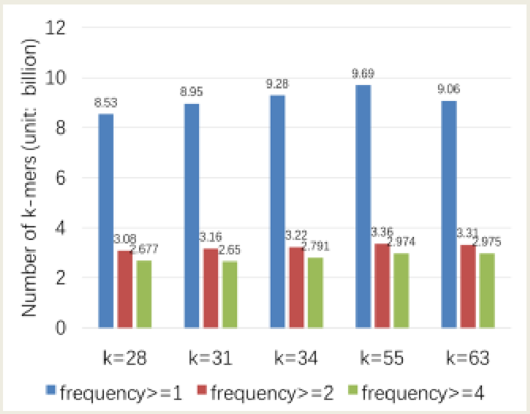
Next-Generation Sequencing (NGS) has caused an unprecedented increase in high-quality sequencing data.In several bioinformatics pipelines, kmer (sub-string of length k) counting is an essential stage in the sequence pre-analysis. many types of sequencing errors exist, including substitutions, insertions, and deletions. From Fig. 1, one can see the statistics of Yanhuang data set as a function of the length of kmer.Recently there has been much effort to develop tools for sequence analysis based on kmer counting. The intermediate data size of storing kmers in memory can be dramatically larger than the input data size. The most popular parallel genome assembly tools are based on de Bruijn graph and bloom filter-based tools, such as BFCounter, Turtle, and KHmerCounter.

The scalability of existing kmer counting tools with distributed memory is minimal. On a single computing node with limited resources, distributed tools, especially KMC2, rarely outperformed shared memory tools. Same reliability problems arise in the I/O and connectivity parts of the process

Fig. **1** The effect of sequencing error on distinct kmer's frequency distribution of Yanhuang data set. The kmers with its frequency less than and equal to 3 can be recognized as kmers generated by sequencing error .

SWAPCounter is the first kmer counting method that scales to 32,768 cores. It also outperforms state-of-the-art single node kmer courting methods, such as KMC2. and MSPKmer Counter. Our analysis makes three contributions:

1. An MPI streaming I/O module is equipped with caching mechanisms and data pooling techniques.
2. Nonblocking all-to-all connectivity is used to overlap computing and communication.
3. As an optional step in the kmer filtering process, we use a counting bloom filter to exclude low abundance kmers.



The rest of the paper is organized as follows. Section **2** gives a brief introduction on kmers and counting bloom filter. Section **3** describes the design and algorithm of SWAPCounter. Section **4** shows Platform and Data Sets for Evaluation. Section **5** presents and analyzes previous works on kmer counting. Section **6** summarises the conclusion.