ACCELERATING K-MER COUNTING FOR GENOMIC ANALYSIS

Gunavaran Brihadiswaran

(208032P)

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

June 2021

DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other media. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: *UOM Verified Signature* Date: 10.10.2021

The above candidate has carried out research for the Master's thesis under my supervision.

Name of the supervisor: Prof. V. S. D. Jayasena

Signature of the superviso *UOM Verified Signature* Date: 13/10/2021

Abstract

k-mer counting is the process of counting *k* length substrings in a sequence. It is an important step in many bioinformatics applications including genome assembly, sequence error correction, and sequence alignment. Even though generating *k*-mer histograms seems simple and straightforward, processing large datasets efficiently with limited resources, especially memory, is very challenging. As the advancements in next-generation sequencing technologies have resulted in a tremendous growth of genomic data, it is inevitable for *k*-mer counters to be faster and more efficient. A lot of work has been done in the past decade to optimize *k*-mer counting.

Frigate, a fast and efficient tool capable of counting and querying *k*-mers is presented. Its in-memory design utilizes multithreaded, lock-free data structures to improve performance. Thread synchronization is handled using the compare-and-swap technique. The parallel processing pipeline of Frigate is the result of careful performance engineering and design. Frigate was developed with the emphasis on values of *k* less than 20, aiming to maximize performance by employing different algorithms for different ranges of *k* values.

The performance of Frigate was compared with six state-of-the-art *k*-mer counters: Jellyfish, DSK, Gerbil, CHTKC, KMC2, and KMC3, using two real-world datasets. The experiments were carried out for *k* values of 10, 15, and 17 using a different number of threads in the range [1, 32]. The results show that Frigate achieves a comparable performance or up to 2-3x speedup compared to its competitors, especially for large datasets. The *k*-mer counters were analyzed based on the running time, amount of memory used, and scalability. The correctness of Frigate was evaluated by comparing the *k*-mer frequency histogram with those of other *k*-mer counters.

Frigate is written in C and freely available at https://github.com/Gunavaran/frigate under MIT license.

Keywords: *K*-mer counting, Genome analysis, Performance engineering, Parallel computing

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor Prof. Sanath Jayasena for giving me the opportunity to be involved in this research and for his invaluable support and guidance throughout this research. I am always thankful for his efforts and kindness when we were struggling to acquire research grants.

I am also thankful to Systems Engineer Mr. Randika Pathirana and all the staff of the Department of Computer Science and Engineering at the University of Moratuwa who helped me a lot in getting access to the university resources.

Last but not least, I would like to thank my parents and friends for their immense support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DNA	Deoxyribonucleic Acid
NGS	Next Generation Sequencing
GPU	Graphics Processing Unit
I/O	Input/Output
SSD	Solid State Drive
DRAM	Dynamic Random Access Memory
FPGA	Field-Programmable Gate Array
NVM	Non-Volatile memory