Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 70 – 79

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

n-step FM-Index for faster pattern matching

Alejandro Chacón*, Juan Carlos Moure, Antonio Espinosa, Porfidio Hernández

Computer Architecture and Operating Systems Department, Escola d'Enginyeria, Universitat Autònoma de Barcelona Campus UAB, Bellaterra 08193, Spain

Abstract

Fast pattern matching is a requirement for many problems, specially for bioinformatics sequence analysis like short read mapping applications. This work presents a variation of the FM-index method, denoted *n*-step FM-index, that is applied in exact match genome search.

We propose an alternative two-dimensional FM-index structure that allows backward-search navigation giving steps of *n* symbols at a time. The main advantages of this arrangement are the reduction of the computational work, but most importantly, the reduction by *n* of the chain of dependent data accesses, and the increase in the temporal locality of the data access pattern. This benefit comes at the expense of increasing the total amount of data required for the index.

We present an in-depth performance analysis of a multi-core implementation of the algorithm using large references (up to 1.5G). We identify memory latency as the major performance limiter for single-thread execution and memory bandwidth for multi-thread execution. Our proposal provides speedups ranging from 1.4× to 2.4×, when there is no limitation on DRAM capacity.

We also analyse the trade-off of compacting the proposed data structure in order to reduce memory capacity requirements, now at the expense of increasing execution time. An extra 33% of DRAM space allows our proposal to improve performance by 1.2×, while doubling DRAM size enables an additional 1.5×.

Our proposal of *n*-step algorithm provides an alternative for pseudo-random memory access algorithms to be redesigned to scale in current and future computer systems.

Keywords: Pattern Matching; FM-index; Parallel Algorithms; Performance Analysis; Multicore Processors;

1. Introduction

Next-generation DNA sequencing technologies produce over 600 gigabases of DNA sequencing reads from a single instrument run. Sequence alignment is required for downstream data analysis. Resequencing projects need to align (most commonly known as mapping) a long list of short sequencing reads to a reference genome, usually with a low number of differences from this reference. For the case of *de novo* sequence assembly, reads must be aligned to other reads in order to recreate a genome. This large amount of data to process demands fast and efficient pattern matching algorithms. Recent sequence alignment software tools, like *Bowtie*[1], SOAP[2], BWA[3], and GEM[4], use a relatively new suffix tree-based algorithm based on the Burrows-Wheeler Transform (BWT)[5] that is called FM-index[6].

*Corresponding author. Tel.: +34-93-581-1990 ; fax: +34-93-581-2478.
E-mail address: {alejandro.chacon, juancarlos.moure, antonio.espinosa, porfidio.hernandez}@uab.es.

Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.170

FM-index implements a backward search mechanism on top of BWT, which allows finding exact pattern matches in a number of steps that is linear with the length of the searched pattern, and independent from the size of the reference sequence. In addition to its good computational complexity, it achieves high compression ratios, allowing to index the full human genome into less than 1.5 GB of memory space.

Improving execution throughput on current computer systems, composed of multiple sockets of multicore CPUs and a hierarchical memory architecture, relies on finding parallelism in the application algorithm and performing data accesses with high reference locality. Temporal locality in the data access stream is exploited to reduce access latency and save both memory bandwidth and energy consumption. Spatial locality is used to amortize costly data accesses to DRAM in the form of read and write bursts. Finally, memory-level parallelism helps tolerating high memory access latencies.

Sequence alignment algorithms usually offer plenty of thread and memory-level parallelism, easily obtained by performing multiple simultaneous pattern searches (one for each short read sequence) on the same reference sequence. This parallelism allows exploting all the execution cores in the system and can be combined with multithreading and prefetching techniques to better exploit the available memory bandwidth. The FM-index search algorithm, though, lacks both temporal and spatial data access locality. Its good properties of high compression and fast operation are at the cost of a pseudo-random data access pattern. This lack of locality creates severe difficulties to make an efficient usage of the memory hierarchy, then limiting its potential scalability.

In order to improve the search scalability, we propose a generalization of the FM-index search algorithm, called *n*-step FM-index. It basically divides the number of search steps by *n*, at the expense of increasing the total amount of data required for the index. Each search step is computationally more complex and the whole amount of data read on each search operation is barely reduced, but the spatial locality of memory accesses is improved, and this also improves performance.

Our main contributions can be summarized as follows:

- We introduce an algorithmic variation of the FM-index search that reduces the number of search steps in order to improve execution time, at the expense of increasing the size of the BWT data structure.
- We present an in-depth analysis of performance of a multi-core implementation of the algorithm using large indexes (up to 1.5 G). Speedups from 1.4× to 2.4× are reached. We identify memory latency and memory bandwidth as the major performance limiters.
- We analyse the trade-off of compacting the BWT data structure to reduce memory capacity requirements at the expense of increasing execution time. An extra 33% of DRAM space allows our proposal to improve performance by 1.2×, while doubling DRAM size enables an additional 1.5×.

In section 2 we are going to describe the FM-index structure and operation. Section 3 introduces our proposal of increasing data locality by n-step FM-index search. In section 4, we provide experimentation results for the execution of the proposal on a multicore system. Section 5 discusses related work and, finally, section 6 summarizes results obtained and future work.

2. Background: single-step Ferragina-Manzini Index (FM-Index)

Indexing a reference sequence or string is a method to accelerate pattern search. The time spent on creating the index is amortized when a large enough number of searches is presented. Total memory capacity requirements to store this index must also be considered. As mentioned above, FM-index is the preferred indexing method used in most sequence alignment software tools. Next, we introduce the fundamental concepts behind the FM-index structure and operation.

2.1. Notation and basic concepts

Let $S = S_{[0]}S_{[1]}\cdots S_{[|S|-1]}$ be a sequence or string over an alphabet Σ, where $S_{[i]}$ is the i^{th} symbol of the string. $S_{i,j} = S_{[i]}S_{[i+1]}\cdots S_{[j]}$ is a *substring* of S. $S_i = S_{[i]}S_{[i+1]}\cdots S_{[|S|-1]}$ denotes a *suffix* of S starting at position i. Representing DNA only requires the symbols {A,C,G,T}. We use R to denote the *reference* sequence and Q to denote a *query* sequence. We suppose that the length of R ($|R|$) is much higher than the length of Q ($|Q|$). The *exact matching* problem consists of finding all the occurrences of Q into R, i.e., the position of the substrings of R that are equal to Q.