Ain Shams University
Faculty of Computer & Information Sciences
Information Systems Department

Master's thesis entitled

A Distributed Indexing Algorithm for Arabic Search Engines

Submitted as a partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences.

By

Nermine Naguib Jean Sophoclis

B.Sc. in Computer and Information Sciences,
Demonstrator at Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

Under Supervision of

Prof. Dr. El-Sayed M. El-Horbaty
Head of Computer Science Department
Professor of Computer Science

Dr. Mohammad Abdeen
Associate Professor of Computer Science

Faculty of Computer and Information Sciences
Ain Shams University

Cairo 2013

ABSTRACT

In contrast to English search engines, Arabic search engines did not have their fair share in modern studies despite the continuous growth of Arabic Internet users and data. Towards bridging the gap, this thesis presents a novel massively parallel indexing algorithm customized for Arabic documents. Our proposed indexing algorithm takes advantage of the highly parallel architecture of graphics processing units (GPUs) both in creating the final index and in the required preprocessing of the original Arabic documents through stemming and normalization.

Since preprocessing consumes a major portion of the execution time, special attention is given to GPU optimization techniques for enhancing the performance of the proposed GPU-accelerated Arabic stemming algorithm. An empirical study assessing and comparing the effect of combining these optimizations is presented in the thesis.

On the other hand, parallelizing the creation of the final index presents an important challenge due to the minimal synchronization support provided by the GPU architecture. We propose novel synchronization techniques including carefully-designed data structures allowing the GPU threads to efficiently collaborate in creating a correct global index.

OpenCL was used as the implementation framework and the tests were run on an NVidia Geforce 310M GPU. The preliminary tests of our GPU-accelerated Arabic indexer showed promising speed-up factors.

SUMMARY

Indexing aims to generate a global web index that helps in finding pages that matches a user's query. In general, indexing deals with huge sizes of web data. Many challenges face the indexing process; for example, the generated index must be moderately sized while guaranteeing mostly correct lookup results at the same time. This is achieved through time-consuming text preprocessing techniques such as stemming and normalization.

In this thesis, we exploit the massively parallel GPU architecture to speed-up the indexing of Arabic documents. Specifically, the main contributions of this thesis are: 1) a novel GPU-accelerated algorithm for indexing Arabic documents including a GPU-accelerated Arabic preprocessor and a GPU-accelerated index generator, 2) A comparative study of various GPU optimization techniques and their combinations, and 3) novel synchronization techniques and suitable data structures allowing GPU threads to collaborate in creating a redundancy-free unified global index.

We did implement the proposed algorithms and their variations using OpenCL and tested them on an NVidia Geforce 310M GPU. The preliminary tests of our GPU-accelerated Arabic indexer showed promising speed-up factors. In the thesis we provide a more detailed account of the tests and their results discussing the strengths and weaknesses of our GPU indexing approach.

ACKNOWLEDGMENTS

My deepest gratitude goes first to God who allowed me to be in this position and guided me along this long journey and blessed all my work. Secondly, it goes to my parents and my husband for their support and encouragement during the period of my work. Their eagerness to get the best out of me is really what made me reach this point.

Throughout the years many people have directly and indirectly helped me achieve this goal. I would like to thank them all, but there are some people who need special recognition.

First of all, I would like to thank my supervisors **Prof. El-Sayed M. El-Horbaty**, and **Dr. Mohammad Abdeen** for their guidance and help. I am extremely grateful to Prof. Sayed for guiding me with his advices which I learned a lot from, and also for his patience. I also want to express my gratitude to Dr. Mohammad for always giving me the time and support I needed during my work, and of course for his technical advices that were very useful to me.

I would also like to thank my parents for their love, care and support my whole life, they are the reason behind all my success. Moreover, I'd like to thank my husband Dr. Cherif Ramzi Salama for his encouragement and support and for being there for me all the time.

Finally, I would like to thank my thesis examination committee for giving me the honor of being my examiners.

Contents

List of Tables **xi**

List of Figures **xi**

1 Introduction **1**
1.1 Search Engines 1
1.1.1 Search Engine Components 1
1.1.2 Introduction to Indexing 3
1.2 General Purpose GPU computing 4
1.3 Thesis Problem Statement 6
1.4 Thesis Organization 7

2 Background and Related Work **9**
2.1 Indexing 9
2.1.1 Indexing Techniques 10
2.1.1.1 Simple Index Construction 10
2.1.1.2 Memory-Based and Disk-Based Inversion . . . 11
2.1.1.3 Two Pass In-Memory Inversion 12
2.1.1.4 Sort-Based Inversion 14
2.1.1.5 Compressed Sort-Based Inversion 15
2.1.1.6 Multi-Way Merge Inversion 16
2.1.1.7 In-Place Multi-Way Merge Inversion 17
2.2 General Purpose GPU Programming With OpenCL 18
2.2.1 OpenCL Architecture 22
2.2.2 Nvidia Geforce 310M GPU 24

3 GPU-Accelerated Light Stemmer For The Arabic Language 26
3.1 Text Preprocessing 26
3.2 Stemming Algorithm 27
3.3 Implementations and Optimizations 29
3.3.1 Arabic Language and Text Processing Challenges . . . 29
3.3.2 Basic GPU Parallel Arabic Light Stemmer Implementation 30
3.3.3 Constant Memory Optimization 32
3.3.4 Block by Block Optimization 32
3.3.5 Global Memory bandwidth optimization 33
3.3.5.1 Copying each word to be stemmed to the private memory 33
3.3.5.2 Minimizing the amounts of data transferred to the GPU 33
3.3.5.3 Using the explicit subdivision mode 34
3.3.6 Pinned Memory Optimization 34
3.4 Results 37
3.4.1 Document by Document Implementations 38
3.4.2 Block by Block Implementations 39
3.4.3 More Test Samples 40

4 Indexing Arabic Documents Through GPU Computing 43
4.1 Algorithm 43
4.1.1 Global and Local Index Data Structures 49
4.2 Synchronization Challenges 53
4.3 Results 54
4.3.1 Results discussion 57

5 Conclusion and Future Work 59
5.1 Conclusion and Contributions 59
5.2 Future Work 60

A GPU Stemmer Details and Pseudo-code 62
A.1 Arabic GPU Stemmer's Device Pseudo-Code 62
A.1.1 Stemming Algorithm Summarized 62

A.1.2 Suffixes and Prefixes Removal Rules 63
A.1.2.1 Suffixes Removal 63
A.1.2.2 Prefixes Removal 63
A.1.2.2.1 Phase 1 63
A.1.2.2.2 Phase 2 63
A.1.3 GPU Stemmer Pseudo-Code 64
A.1.3.1 Definitions 64
A.1.3.2 Notes 64
A.1.3.3 GPU Stemmer Kernel 64
A.1.3.4 Notes 65
A.1.3.5 removeSuffixes Function 67
A.1.3.6 Notes 68
A.1.3.7 removePrefixes Function 69
A.1.3.8 Notes 69

References **70**

Publications **80**

List of Tables

2.1 Brief Comparison between Multicore and Many-Core microprocessors 19
2.2 Nvidia Geforce 310M GPU specifications 24

3.1 The stemming kernel input/output arguments 31
3.2 The execution time of the document-by-document approaches and the sequential approach. 35
3.3 The execution time of the block by block approaches and the sequential approach 36
3.4 The speed-up factors of the block by block approaches over the sequential approach. 36
3.5 The execution time of the block by block with constant and pinned memory approach and the sequential approach. 41
3.6 The speed-up factors of the block by block with constant and pinned memory approach over the sequential approach 42

4.1 Kernel and overall execution time of the GPU and CPU indexers 55

List of Figures

1.1 General Search Engine Components (Source: "http://www.solutionhacker.com/powerful-full-text-search-engine-lucene-part-1/") . . . 3
1.2 Types of indexes . . . 5
1.3 Inverted Index Structure . . . 5

2.1 In-Place Merging (adapted from "Managing Gigabytes" [1]) . . . 17
2.2 Conceptual OpenCL device architecture with processing elements (PE)and compute units. The host is not shown. (Taken from "The OpenCL Specifications" page 25 [2]) . . . 23

3.1 Flow chart of the stemming algorithm adopted in our work. . . . 28
3.2 Comparing the overall execution time of the document-by-document approaches with the sequential approach. . . . 35
3.3 Comparing the execution time of the block by block approaches with the sequential approach. . . . 36
3.4 Comparing the execution time of the block by block with constant memory technique with the sequential approach. . . . 41
3.5 The speed-up factors achieved by the block by block with constant and pinned memory approach over the sequential approach. 42

4.1 Workflow of the GPU Arabic indexer . . . 45
4.2 Pseudo code of Arabic GPU indexer . . . 46
4.3 Snapshot of the TokensBuffer array . . . 47
4.4 Pseudocode for adding a word stem to the local index hashtable 50
4.5 Pseudocode for adding a word stem to the global index hashtable 51
4.6 Kernel and overall speed-ups . . . 55

Chapter 1

Introduction

1.1 Search Engines

Search engines have been developed long ago for the sake of helping internet users to search and browse web pages. Search engines are responsible for organizing, managing, and storing information about web pages that it already fetched from the World Wide Web. A search engine consists of two main parts;

- Off-line part: where the search engine fetches web pages, and builds an index for their contents.
- On-line part: where it processes users queries and responds to them [3].

1.1.1 Search Engine Components

Generally, the common components of search engines, as shown in Figure 1.1 are:

- **Web crawler:** The Web crawler is responsible for following every link on a web page. It's a mean for providing search engines with up to date data. Crawlers keep a copy of visited web pages to be further processed and indexed. Web crawlers have to behave according to certain policies like: selection policy that states which pages to download, re-visit policy that states when to check for changes to the pages, politeness

policy that states how to avoid overloading Web sites, and parallelization policy that states how to coordinate distributed Web crawlers.

- **Indexer:** The indexer is responsible for storing information (index) about web pages that facilitates the information retrieval. It helps in increasing the performance and speed of finding relevant documents for search query.

- **Query Engine:** The query engine is responsible for managing user queries. A query engine parses the user query then lookups the index and provides a listing of best-matching web pages. Most search engines support the use of the Boolean operators AND, OR and NOT to narrow the search query.

- **Page Ranking module:** The page ranking module is responsible for defining and ordering (prioritizing) the best matching results to be shown to the user first. It gives ranks according to link analysis algorithms. It gives weights to each element of a hyperlinked set of documents "measuring" its relative importance within the set. Some of the factors that affect the page rank of a web page are: the quantity of inbound links (links pointing to this page), the page rank of the pages providing the links, and the relevance of search query words on the page [4].

Web search engines work by storing information about many web pages, which its crawler (spider) retrieves. The contents of each page are then analysed to build an index for them. When a user enters a query into a search engine, the query engine parses it and examines the index for the best matching web pages. Page Ranking is used to order the best matching pages by their relevance to the search words. Documents appearing on top of the results are considered to be more relevant to the user's search.

The usefulness and effectiveness of any search engine depends on "Precision (i.e. number of relevant documents retrieved over the total number of retrieved documents) and Recall (i.e. number of relevant documents retrieved over the total number of relevant documents in the collection)" [5]. However, Users are much considered by the response time and the large collections of

web pages available in the search engine's index [3]. These criteria imply the necessity of high performance indexing specially that the web data and users are in a huge continuous increase. In this thesis we are concerned with increasing the performance of indexing specially for the Arabic language.

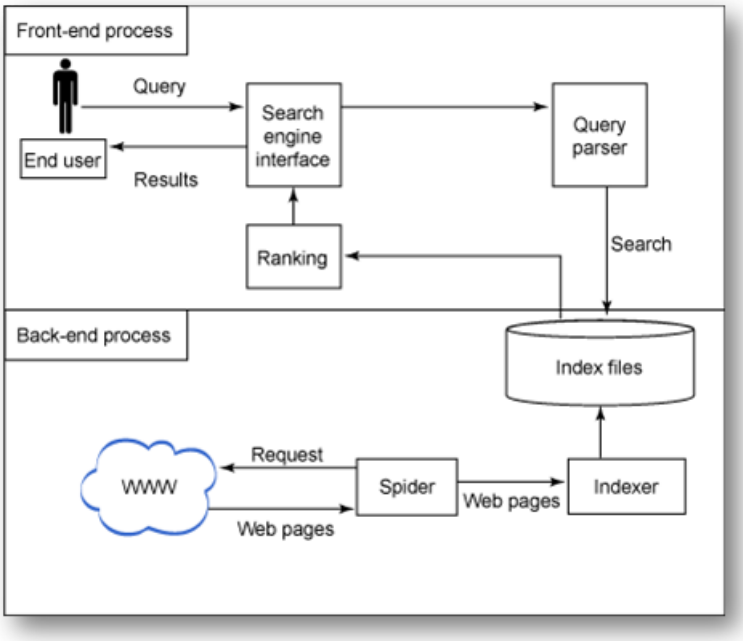

Figure 1.1: General Search Engine Components (Source: "http://www.solutionhacker.com/powerful-full-text-search-engine-lucene-part-1/")

1.1.2 Introduction to Indexing

Indexing is the core of any search engine. The aim of indexing is mainly to create an index (similar to that of a book) for all the terms occurring in a set of documents recording where they appeared along with their frequency of occurrence.

There are many types of indexes. Figure 1.2 summarizes the different types of indexes. This work is concerned with Text Index which has four types:

- Document-term matrix, which stores the occurrences of words in documents in a two-dimensional sparse matrix.
- Suffix tree which is an array of integers giving the starting positions of suffixes of a string in lexicographical order.
- Inverted index which stores a list of occurrences of each unique term, typically in the form of a hash table or binary tree.
- Signature files; the idea behind signature files is to create a quick filter that will keep all the documents that match the query and hopefully a few ones that do not. The way this is done is by creating for each file a signature, typically a hash coded version.

In our work we focus on inverted file indexing as it was proven to be the most practical form of indexes [6, 1, 7]. Figure 1.3 gives a demonstration of the inverted file index structure. An inverted file index consists of two major components.

- The search structure or vocabulary stores for each distinct word t
  - A count f_t of the total occurrences of word t in all the documents containing it, and
  - A pointer to the start of the corresponding inverted list (Postings list).
- The second component of the index is a set of inverted lists where each list stores for the corresponding word t
  - The identifiers d of documents containing t, represented as ordinal document numbers, and
  - The associated set of frequencies $f_{d,t}$ of terms t in document d

1.2 General Purpose GPU computing

General purpose computing on the graphics processing unit (GPU) can be defined as the process of running variety of general purpose programs on the

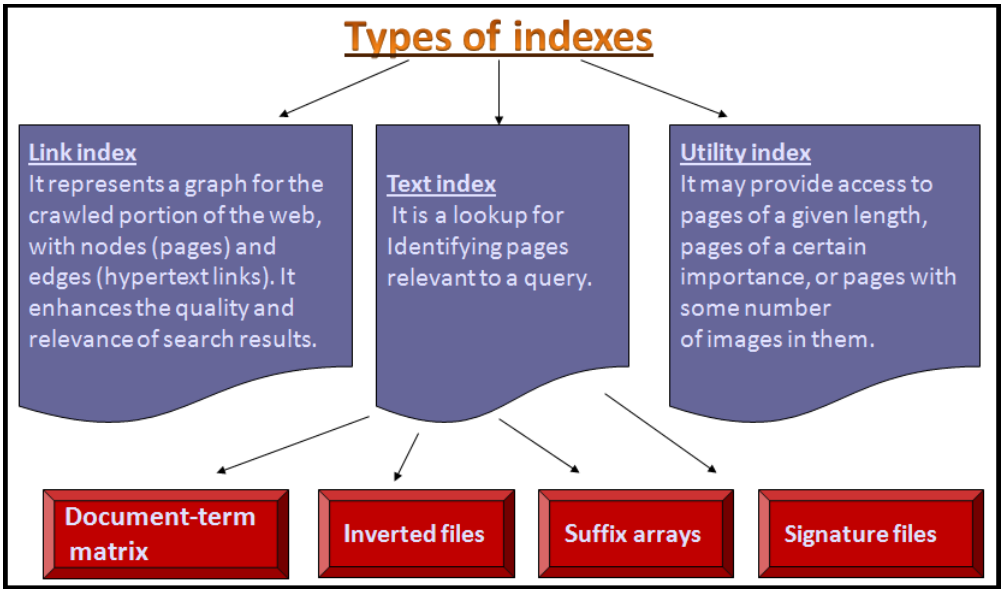

Figure 1.2: Types of indexes

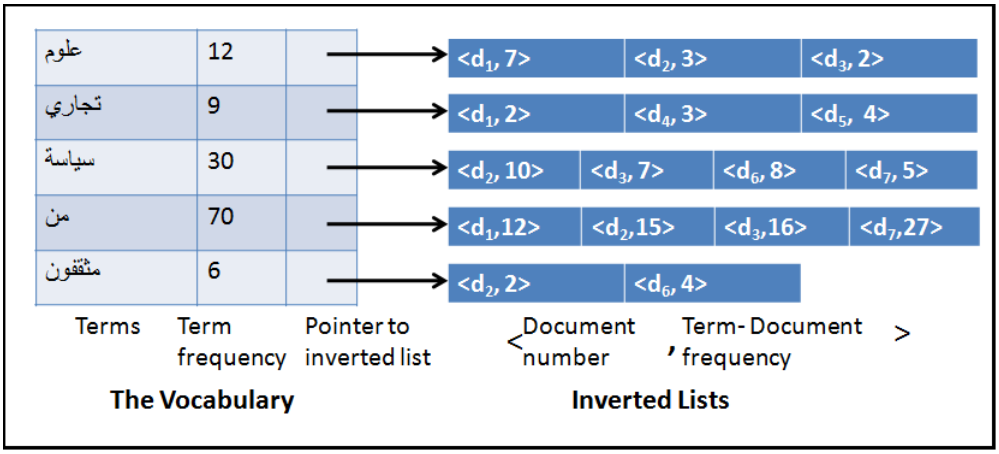

Figure 1.3: Inverted Index Structure

GPU taking advantage of its highly parallel architecture (i.e; manycore architecture as explained in Chapter 2) to run tasks other than those of graphics rendering. Since general purpose GPU programming was introduced, it made possible for ordinary desktop computers and laptops to run massively parallel programs that previously needed computers with much advanced powers. This trend opened the door for the development of various applications in different fields, such as business, biology, science, engineering, etc... to run in parallel on the GPU achieving high performance.

In this work we present a novel indexing approach customized for the Arabic language that runs on the GPU. Our approach exploits the technology of many core highly parallel GPUs to increase the performance of Arabic indexing rather than running on several machines. However, the parallel environment of the GPU poses several challenges related to synchronization. In our implementation we present solutions to achieve synchronization without sacrificing the performance as much as possible.

We implemented the proposed algorithm using OpenCL and tested it on an NVidia Geforce 310M GPU whose specifications will be demonstrated in the next Chapter. The preliminary tests of our GPU-accelerated Arabic indexer showed promising speed-up factors.

1.3 Thesis Problem Statement

The objective of this thesis is to investigate the problem of enhancing the performance of the one component in the Arabic language processing chain, namely the "indexer", through employing general purpose GPU parallel processing. In particular, the thesis investigates the problem of designing a parallel indexer capable of dealing with huge sizes of web data creating a moderately sized index guaranteeing mostly correct lookup results. This requires Arabic-specific preprocessing such as stemming and normalization algorithms, whose performance are also to be enhanced through distributing their workloads over GPU cores. An equally important goal of this thesis is investigating how the GPU indexer could overcome the synchronization challenges arising from the need to access the same local/global index by multiple GPU cores simultaneously during the indexing process.