KARADENİZ TECHNICAL UNIVERSITY
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

COMPUTER ENGINEERING GRADUATE PROGRAM

ACCELERATING GENE IDENTIFICATION IN DNA SEQUENCES WITH CUDA AND OPENCL

MASTER THESIS

Shafiei Abdi SULEIMAN

JULY 2019
TRABZON

KARADENİZ TECHNICAL UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

COMPUTER ENGINEERING GRADUATE PROGRAM

ACCELERATING GENE IDENTIFICATION IN DNA SEQUENCES WITH CUDA AND OPENCL

Shafiei Abdi SULEIMAN

This thesis is accepted to give the degree of

"MASTER OF SCIENCE IN MINING ENGINEERING"

By

The Graduate School of Natural and Applied Sciences at
Karadeniz Technical University

The Date of Submission : **16 / 05 /2019**

The Date of Examination : **30 / 07 /2019**

Supervisor : **Asst.Prof Dr. İbrahimn SAVRAN**

Trabzon 2019

KARADENİZ TECHNICAL UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

COMPUTER ENGINEERING GRADUATE PROGRAM

Shafiei ABDI SULEIMAN

ACCELERATING GENE IDENTIFICATION IN DNA SEQUENCES WITH CUDA AND OPENCL

Has been accepted as a thesis of

MASTER OF SCIENCE

after the Examination by the Jury Assigned by the Administrative Board of the Graduate School of Natural and Applied Sciences with the Decision Number **1808** dated 25 / 06 / 2019

Approved By

Chairman : **Prof. Dr. Abdulsamet HAŞILOĞLU**

Member : **Asst. Prof. Dr. Ibrahim SAVRAN**

Member : **Asst. Prof. Dr. Hüseyin PEHLİVAN**

Prof. Dr. Asim KADIOĞLU

Director of Graduate School

ACKNOWLEDGMENT

First and foremost, praises and thanks to the God, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.

I would like to express my deepest thanks and gratitude to my supervisor Dr. Ibrahim SAVRAN for support, guidance, encouragement and valuable advice and discussions throughout the progress of this thesis. No doubt, without his valuable effort, it would be much more difficult for me to reach my goals. Any words of gratitude would not be sufficient for expressing my feelings.

Also, I would like to express my deep gratitude to Turks Abroad and Related Communities (YTB) for giving me the opportunity to come and study in Turkey.

I would also like to thank my family who made all the sacrifices and supported me throughout this journey. Your prayers have helped me sustain this far.

Lastly would like to thank my friends for their encouragement support specially Nawaal Abdullah and Saleebaan Mohamed for their inspirations at my frustration time regarding my thesis completion.

Shafiei Abdi Suleiman

Trabzon 2019

THESIS STATEMENT

I declare that the research work in this thesis entitled “Accelerating Gene Identification in DNA Sequences with CUDA and OpenCL” is a record of an authentic research work carried out by me, Shafiei Adi Suleiman under the guidance of my supervisor Dr. Ibrahim SAVRAN. All data used in this master thesis are obtained by simulation and experiment work, which I obtained by following all research rules. For the award of any degree or diploma, neither this thesis nor any part of it has been submitted to the any other University of Institute, except where due reference acknowledgment has been given in the text. 30/07/2019

Shafiei Abdi

CONTENTS

Page No

Master Thesis

SUMMARY

ACCELERATING GENE IDENTIFICATION IN DNA SEQUENCES WITH CUDA AND OPENCL

Shafiei Abdi Suleiman

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Computer Engineering Graduate Program
Supervisor: Asst. Prof. Dr. Ibrahim SAVRAN
2019, 37 Pages

Metagenome is the genomic data obtained from the environment. It may contain hundreds or even millions of different species. The current generation sequencing devices can produce billions of sequences at once. Applications such as MGC and Orphelia were initially used for the detection of the genes within those sequences that have been produced by the genetic sequencing devices are now insufficient. Such applications that are not equipped to process billions of sequences could take several days for the gene identification once the sequence file is divided into small divisions up to 20K sequence. In this work, the use of Metagenome Gene Caller (MGC) application for the development of metagenome gene detection has been used. The core functions have been prepared for the OpenCL and CUDA platforms through the appropriate utilization of the data transformations. Furthermore, these core functions were run through the GPU and the results are debated accordingly. The entire gene detection method has been reduced from several days to a few minutes through data structure simplification.

Key Words: Computational Genomics; Next generation Technology; Bioinformatics; Metagenome Gene Caller-MGC; High Performance Computing; MetaGene; Orphelia; FragGeneScan; CUDA; OpenCL, Graphic Processor Unit.

Yüksek Lisans Tezi

ÖZET

DNA SEKANSLARINDA GENLERIN CUDA VE OPENCL KULLANILARAK HIZLI SAPTANMASI

Shafiei Abdi Suleiman

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Dr. Öğr. Üye. Ibrahim SAVRAN
2019, 37 Sayfa

Metagenom, ortamdan elde edilen genomik veridir. Yüzlerce, hatta milyonlarca farklı tür içerebilir. Mevcut nesil sıralama cihazları bir kerede milyarlarca dizi üretebilir. MGC ve Orphelia gibi uygulamalar, başlangıçta, genetik dizilim aygıtları tarafından üretilen dizilerdeki genlerin tespiti için kullanılmıştır. Milyarlarca diziyi işlemek için donatılmamış bu tür uygulamalar, dizilim dosyası 20K dizisine kadar küçük bölümlere bölündüğünde gen tanımlaması için birkaç gün sürebilir. Bu çalışmada, metagenom gen saptamasının geliştirilmesinde Metagenom Gen Arayan (MGC) uygulamasının kullanımı kullanılmıştır. Temel fonksiyonlar, OpenCL ve CUDA platformları için veri dönüşümlerinin uygun şekilde kullanılmasıyla hazırlanmıştır. Ayrıca, bu çekirdek işlevler GPU'da çalıştırıldı ve sonuçlar buna göre tartışıldı. Veri yapısı basitleştirmesiyle tüm gen saptama yöntemi birkaç günden birkaç dakikaya indirilmiştir.

Anahtar Kelimeler: İşlemsel Genomik; Yeni Nesil Teknoloji; Biyoinformatik; Metagenom Gen Arayan-MGC; Yüksek Performanslı Bilgi İşlem; MetaGene; Orphelia; FragGeneScan; CUDA; OpenCL, Grafik İşlemci Birimi.

LIST OF FIGURES

Page No

LIST OF TABLES

Page No

ABBREVIATIONS

ANN	Artificial Neural Network
CUDA	Compute Unified Device Architecture
CPU	Central Process Unit
NGS	Next Generation Sequence
GPU	Graphical Processing Unit
DNA	Deoxyribonucleic Acid
RNA	Ribonucleic acid
A	Adenine
G	Guanine
C	Cytosine
T	Thymine
MGC	Metagenomic Gene Celler

1. INTRODUCTION

Conventional approaches to the isolation and microbe culturing had been unsuccessful in the determination of a diverse community of the microbes. Only 1-10 per cent of the total microbial species could be cultured [1, 2, 3, 4]. The modern approach entails accessing the plethora of genetic information with the specific use of the environmental DNA extraction that offers a mean to avoid the limitations associated with culture-dependent exploitation of the genetic material.

Introduction of the pioneer research experimentation conducted by Sanger and Coulson till date, sequence analyses have been given utmost importance in congruence with molecular biology. Research in the past thirty years has successfully depicted numerous studies projecting various genome species with functional and structural annotations.

The technology involving sequencing has significantly progressed since 2007 with the invention of the technologies facilitating lowering per-base cost while improving the throughput. For instance, the Human Genome Project (HGP) cost more than $3 billion in 2001 and also requires ten years to complete. However, the in-process revolution encapsulating next-generation sequencing (NGS) technologies has promoted to the generation of high-throughput short read (HTSR) technique at relatively less costly compared to the traditional sequencing technologies. the Next Generation Sequencing (NGS) holds the potential to sequence a human-sized genome within a day in less than $1000.

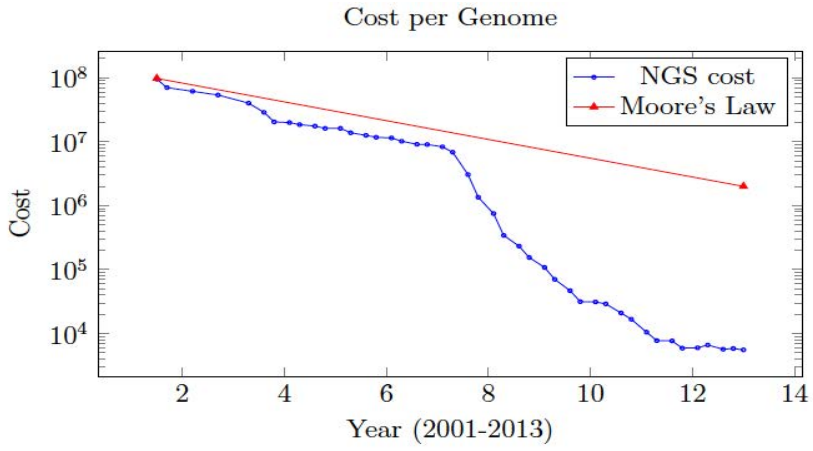

Figure 1 : Cost per Genome: the cost of sequencing a human-sized genome(5,6,7)

As illustrated in Fig 1., a decrease in the cost of DNA sequencing in the past 10 years concerning Moore's Law. Y-axis represents a logarithmic scale within the graph. Prompt and rapid Moore's Law out-pacing is evident after 2007. Nevertheless, there are several challenges linked to designing a computational solution to analyze the NGS data due to the following reasons:

1. Large Sequence Data: Increasing rate of the cultivation of biological sequences with the use of NGS technology has promoted in the intense growth of publicly easy accessible sequence data sets. Processing a large number of data tends to increase run-time requirements and consumes excessive memory.

2. Computational Requirements: Sequence analysis is many times computationally complex based on the different issues. Polynomial solutions require large memory as well as a massive run-time for storing and running large sequences of data. For instance, Alignment represents a computational relationship between two strings. With the use of dynamic programming, O(l2) time is required as the for the computation of the optimal alignment and O(Ll) space. Here, l denotes the string length (l = |s1| = |s2|). Nevertheless, resolving multiple programming programs for dynamic sequence alignments promptly is often intractable. $-O(\binom{n}{2} \times i^2)$

1.1. Thesis Goal

This thesis is interested in examining the acceleration of gene identification in DNA sequences with OpenCL and CUDA. This thesis will specifically compare the two techniques, and analyze the results obtained based on factors such as accuracy, specificity, and sensitivity. Both OpenCL and CUDA are platforms through which Graphics Processing Units (GPUs) can be programmed. CUDA only works on NVIDIA GPUs while OpenCL works with various processors such as Digital Signal Processors, GPUs, CPUs, and others. This thesis assumes that the CUDA platform will be much more accurate than the OpenCL platform, that a particular platform will have better features and will make a little compromise to overcome the open platform.

The thesis is organized as follows. In Section 1. Introduction, In Section 2, Background, Biology overview, DNS Sequences, Whole-genome shotgun sequences, Hierarchal sequences, Next Generation methods, Gene prediction methods, MetaGene,

Orphelia, FregGensscan, MGC, MGC Algorithms, High performance computing, GPU Architecture, Streaming Multiprocessor, GPU Memory model, GPU Programming, CUDA, CUDA Matrix Addition Example, OpenCL, is discussed. In Section 3, Gene Determination in DNA sequence methods is presented. Gene extraction Algorithm from Metagenome is discussed in Section 4, in Section 5 Results discussion are presented. In Section 6, future work is presented.

2. BACKGROUND

2.1. Biology

A cell is an essential unit of life that is often regarded as the "building block of life". Every living entity comprises of single or multiple cells. All cells perform different functions, are different in shapes and sizes. On the contrary, all cells comprise of some similar traits. Every living cell includes a cytoplasm that is enclosed within a membrane. The membrane acts as a boundary separating the interior of the cell with the exterior while also serving as a filter to promote the movement of certain molecules in and out of the cells. Nutrients acquired from the environment helps the cell to grow and enable in the creation of other molecules while also facilitating the interaction with other existential molecules [9]. Cells are also capable to reproduce once they constitute enough components within the original cell to replicate and produce duplicate offspring.

Different cellular processes are promoted by the proteins found within the cells. Proteins comprise of long amino acid chains that are linked through peptide bonds to create polypeptides. Folding of the polypeptides by force within the atoms helps in the creation of protein. Accurate protein form is critical to the functional ability [10]. Different functions of the proteins include breaking down fats, carbohydrates, and other protein molecules, fighting foreign bodies such as bacteria and viruses, and promoting a chemical reaction (where a protein is known as an enzyme) [11]. Enzymes serve as a facilitator to the reagents by lowering the energy of activation for the reaction to enhance its reaction speed. However, during the chemical reaction, enzymes are not consumed and hence are only utilized for catalyzing the reactions [12, 13].

It is also evident that proteins are essential in the sustenance of the cell. All the information that is required for the creation of the protein is stored within the deoxyribonucleic acid [DNA]. DNA is used for translating the new proteins through the process named as genetic expression. Also, DNA helps the cell in reproduction through replication. Gene is defined as a component of the DNA that encodes genetic information. Every DNA molecule comprises of 2 strands or long nucleotide chains that are intertwined to form a double helix. Every nucleotide further constitutes a deoxyribose sugar that forms the nucleobase in addition to a phosphate group [14]. There are four nucleobases which are

Adenine [A], Cytocise [C, Guanine [G], and Thymine [T]. Even though one base in a single strand bond with the opposing bases found in the other strand, A only bonds with T while C bonds with G.

A schematic overview illustrating the DNA components is found in Fig 2.1 while Fig 2.3 demonstrates a chemical structure of different molecules and its integration within a new nucleotide. Every DNA strand has two ends: a 3'-end and a 5'-end (called as 5-primer). 5-primer is responsible for binding with the phosphate group while the 3-primer bonds with a sugar base. This structure is the backbone of the DNA with phosphate groups and altering sugar pairs. Both strands are anti-parallel meaning that one 5-primer is linked to 3-primer. The strand orientation is regarded as Significant which indicates that the translation and the replication could only be preceded with a 5' 3' direction. The replication of DNA is shown in Fig 2.2.

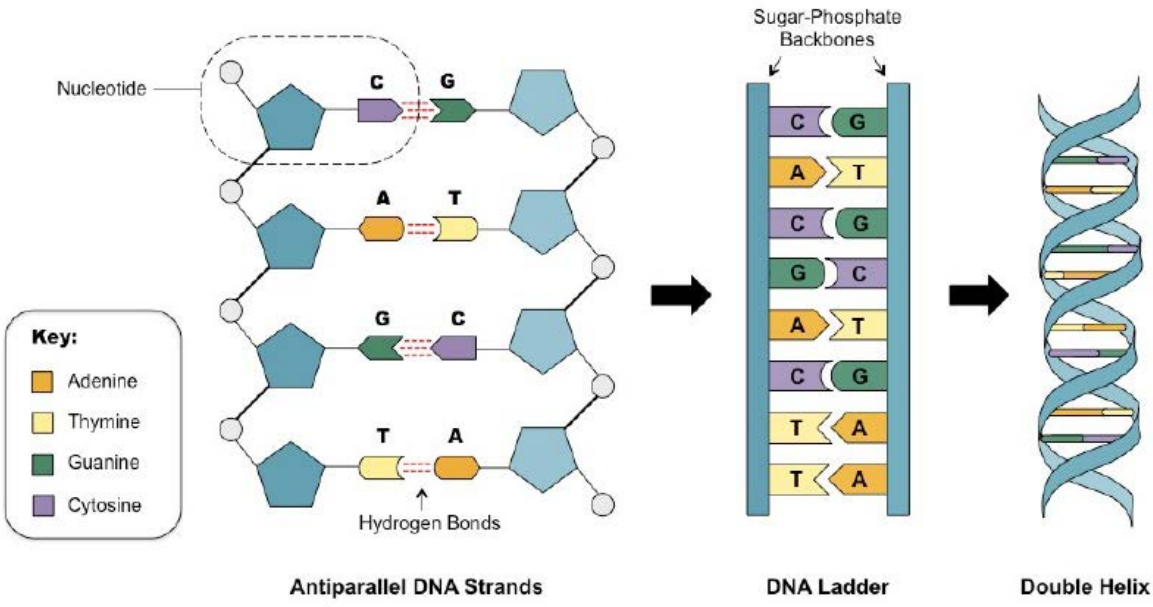

Figure 2. Structure of DNA, source: [15]

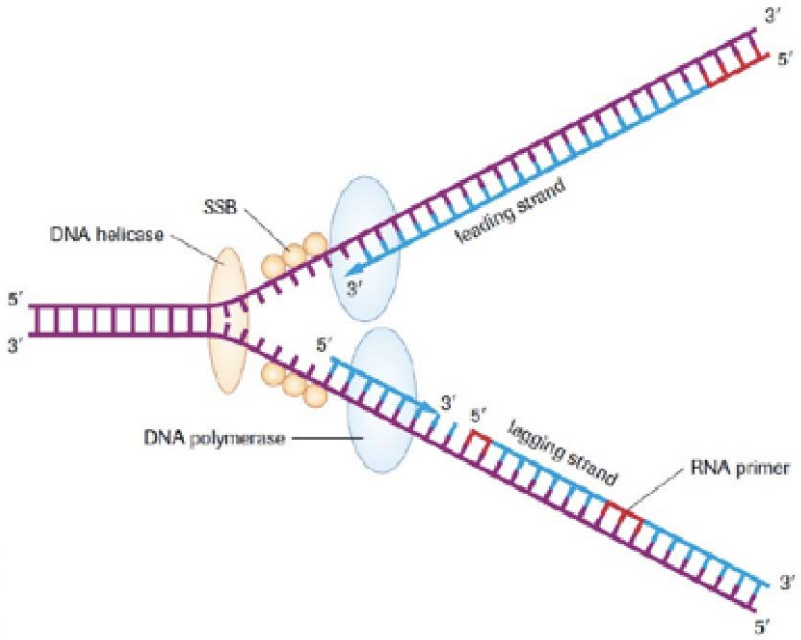

Figure 3. DNA replication, note that the lagging strand is copied in multiple pieces , Source (16)

Strands within the DNA are split during the replication phase enabling the nucleotides to be read. DNA strands are prevented by the single-stranded binding (SSB) from binding with each other (annealing). Different RNA primers are also introduced within the DNA strands where one RNA primer is added within the leading strand while many others are added within the lagging strand. Enzyme DNA polymerase initiates to build from these primers. With only one RNA primer, the leading strand is produced in one go however multiple primers within the lagging strands produces several pieces along with the primers. These pieces are regarded as Okazaki fragments [17]. Ligase is responsible for stitching different fragments of the DNA together after RNA (Ribonuclease) eliminates the primers.

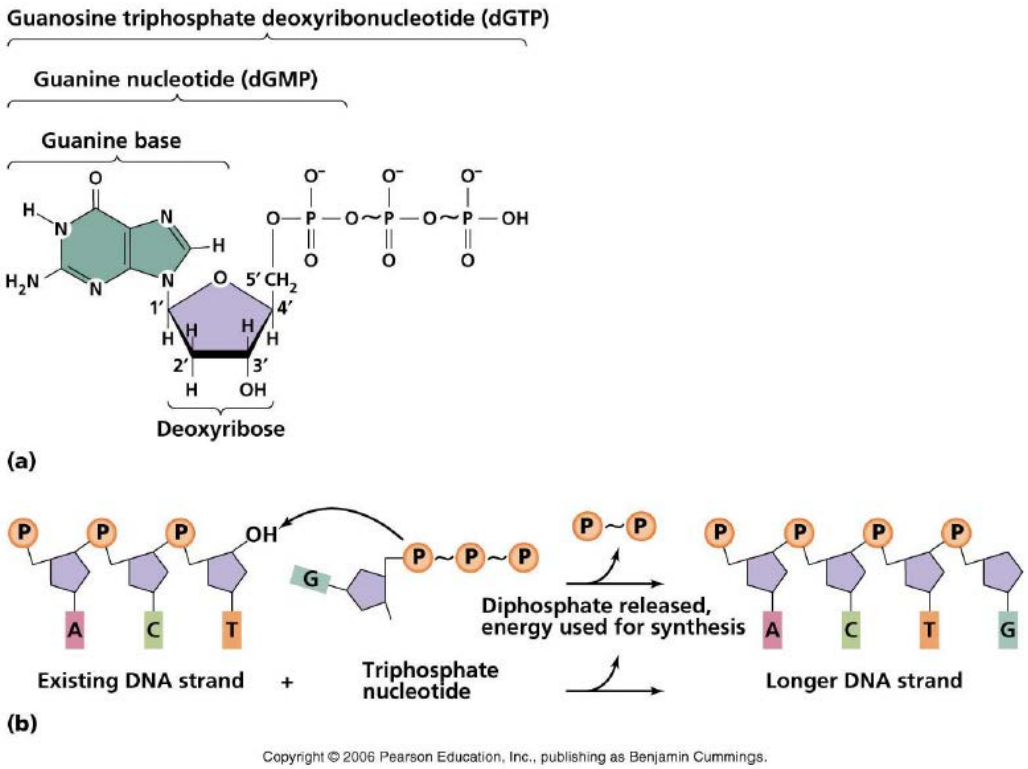

Figure 4. Molecules involved in DNA replication, source [18]

DNA produces new proteins with the help of messenger RNA (mRNA) that is similar to the structure of DNA however it comprises of one strand only. In addition, Thymine base within the RNA is replaced by Uracil (U) and the sugar present within RNA is ribose rather than deoxyribose. Three bases are paired together (commonly known as codon) that helps in DNA reading. Every codon represents an amino acid. However, as there are 20 different kinds of amino acids hence 43 = 64 make deferent codons while multiple codons are assigned to the similar amino acid. Furthermore, codons have special attributes like start (ATG) while (TAG, TAA, TGA) as to stop [19].

As soon as the RNA polymerase enzyme correctly binds to the DNA with the assistance of promotor, it is the indication of the starting of the gene. The strand of mRNA is looking like the coding strand (sense) of the DNA whereas another strand of DNA is identified as the template (antisense) strand. A transcription bubble is generated by the RNA polymerase that is a small part of the separated DNA that allows the RNA polymerase to gain access of the bases within the coding strand. This specific DNA strand is isolated from the template strand thereby promoting the other strands of DNA to align

back together. Hence, the mRNA strand now resembles the coding strand of DNA except the T (thymine)base is replaced by U (uracil).

The strand of mRNA is translated as a protein through the ribosome, which is a very complicated molecule comprising of different ribosomal RNA molecules along with a dozen of proteins [20]. Binding of the ribosome takes place at the start of the mRNA strand. Transfer RNA (tRNA) is responsible for the interpretation of the mRNA codons. For this, tRNA first attaches to the AUG which is a start codon and is always conducted by the methionine amino acid. Ribosomes then keep on searching for other tRNA molecules that rightly fit mRNA codons and adds the amino acids to continue the growth of the chain while producing the protein. The molecules within the tRNA are not consumed and hence could be reused after picking new amino acids. UAG, UAA, and UGA (stop codons) do not fit any tRNA molecules therefore rather than binding to tRNA, another group of protein known as 'release factors' is bound with the mRNA for the ribosomes to release the proteins [21]. Degradation of the used mRNA occurs once they are consumed [22].

Protein comprises of the amino acid chain which is connected by peptide linkages and is known as the primary structure [23]. Since many of the amino acids are non-polar hence they have no electrical charge. As opposed other amino acids either contain a negative or positive charge or no charge yet they contains a dipole so they are known as polar. Hydrogen bonds are formed from the polar amino acids [24] that help in the formation of ionic bonds from the charged amino acids [25]. Weaker Van der Waal bonds are formed by the hydrophobic amino acids. Furthermore, as a large number of bonds are non-covalent therefore there is no sharing of the electrons [25]. Only one covalent bond could be formed by Cysteine [23].

Folding patterns takes place due to bonding in different parts of the proteins. Two types of patterns are commonly formed which include the alpha (α) and the beta (β) helices (See Fig 2.4). These folding patterns constitute the secondary protein structure. Interaction between the patterns due to the presence of Van der Waal forces leads to the formation of the tertiary protein structure. Lastly, a protein comprising of multiple subunits or chains is known as a quaternary structure [23].

Partial folding within the protein enables in the interaction with different molecules within the cell leading to improper folding. Such misfolding in the protein could also combine and with other molecules to form large aggregates. To avoid this, chaperone proteins protect the proteins throughout the folding process. Chaperone proteins are many

times heat shock proteins as heat reduces the stability of the proteins. The cells generate more heat shock proteins once they are exposed to high temperatures [26].

2.2. DNA Sequencıng Methods

Determination of the accurate order of the molecules within a DNA/RNA molecule is called sequencing. Sanger and Coulson [27] proposed a “plus and minus” method in 1975 for DNA molecule sequencing. A similar method was proposed by Sanger et al. in 1977 which was known as the “chain termination method” [28]. Presently, the majority of the sequencing techniques follow the chain termination method. With the inception of this method, increased throughput within the technological advancement has taken place to decrease the per base costing.

Currently, DNA sequencing techniques can sequence almost 100-1000bp nucleotides with a >98 per cent accuracy level. On the other hand, DNA and RNA are much larger genetic molecules comprising of few 10s of 1000s to 10s to millions nucleotide and one protein containing more than hundred chains of amino acids. Phase strategy is the most commonly used technique for assembling the target molecule. This consists of (1) sequence of randomly selected “fragments” from different molecule copies and (2) substantially depending on different computational methods for assembling the targeted molecule’s sequence. I have offered a summarized overview of the multiple sequencing techniques as well as different sequences types which are retrieved from specialized methods and have been discussed in the following section.

2.2.1. Sequencing Through Whole Genome Shotgun (WGS)

WGS is the approach which is implied for sequencing a complete genome. Initially, this technology was applied for sequencing the bacteriophage genome [29]. However, as the “chain termination process” could only be utilized for DNA sequences consisting of relatively shorter sequences (100-1000bp), therefore WGS technique is used as a large chunk strategy that samples random different locations within a target genome to extract short sequences (~5000bp). Furthermore, such sequences are then cloned within the

bacterial vector colonies (BAC) to be sequenced from both sides. Resultant sequences consist of 500-1000bp in length and are known as shotgun fragments.

Within WGS sequencing, reads for multiple overlapping in the targeted genome are gained by conducting different rounds of the similar processes, like each targeted base of the genome is expected to be covered by a predefined number of fragments. This number is commonly known as the "sequencing coverage" and is represented by "X". The total number of sequence that is fragmented via WGS project is determined by the target genome length as well as the desired coverage of the sequence. For instance, 10X coverage of genome comprising of 1 billion base pairs would lead in the production of 14 million fragments with an average length of the sequence assumed to be 700bp. As WGS process is randomized, it is difficult to ensure that all the bases would be covered by one or more fragments. In general, and real-life practice, many of the genome stretches are often uncovered within the sequencing and every uncovered stretch is known as a "sequencing gap" Even though the specification of the high coverage reduces the gap frequency, yet this phenomenon leads to increased sequencing cost.

WGS sequencing is cheaper as compared to other sequencing techniques involving HGP [30, 31, 32].

2.2.2. Hierarchical Sequencing

Within this specific approach, breakdown of the genome occurs into clones consisting of 150-200Kbp that is known as "Bacterial Artificial Chromosome" (BAC). Collection of multiple BACs offers a minimum tiling pathway based on locations that help in determining the genome. Every selected BAC is separately sequenced with the use of a shotgun approach that helps in the creation of multiple short shotgun fragments ranging in between 500-1000bp. This technique is also known as the "clone-by-clone" sequencing due to the presence of the hierarchical strategy. Even though this technique is costly as compared to WGS, it offers additional data to help in the accurate fragment analysis. Different colonies like Fosmids and Yeast Artificial Chromosomes are also included in the hierarchical methods. This method is in the process to sequence multiple complicated eukaryotic genomes found within humans [Consortium (2001)] and maize [NSF (2005)].

2.2.3. Next Generation Technique

The need for a lower sequencing cost has led to the innovation of high-throughput techniques to parallelize the process leading to concurrent sequencing of thousands of genomes. More than one million sequencing processes could be run in parallel in ultra-high-throughput sequencing [33, 34].

NGS technology has been summarized in Table 2.1. Specifications of the NGS technique including Ion/Solid Torrent PGM from Life Sciences, GS FLX Titanium by Roche, and MiSeq and HiSeq by Illumina are discussed. Also, important attributes of the computing approaches are discussed within the table concerning next-generation sequencing. Nonetheless, the initial cost of the equipment has not been mentioned within the table. For instance, Pacific Bio sequencers and Illumina are more costly than the Ion sequencer [35].

Table 1. Comparison of next-generation Sequencing Methods

Method name	Read Length	Accuracy	Read/ run	Time/ run	Cost/Mbp
Single-molecule sequencing Pacific Bio	5500 - 8500bp	99.9 %	400Mbp	30-120m	<$1
Ion Torrent sequencing Ion semiconductor	<400bp	98%	80Mbp	120m	$1
Pyrosequencing 454	700bp	99.9 %	1Mbp	24h	$10
Sequencing by synthesis Illumina	50-300bp	98%	3Bbp	1-10d	5-15¢
Sequencing by ligation	50+50bp	99.9 %	1.4Bbp	1-2w	15¢

SOLiD sequencing					
Chain termination Sanger sequencing	400-900bp	99.9 %	-	3h	>$2000

2.2. Gene Prediction Methods

Shotgun sequence resulting from full genome sequencing is derived from a single clone that enables the annotation and assembling of the genome highly manageable in the cultural microbes. Uncultured genomes are directly taken from the environment for the sampling in the metagenomics.

Next-generation sequencing (NGS) in metagenomics leads to the production of a high number of data as compared to traditional sequencing. Conversely, resultant sequences are both partial and noisy, and more importantly, derived from thousands of multiple species. Hence, the annotation and the assembling of huge metagenomics information are linked to additional complexities. Different techniques have demonstrated promising outcomes with improved efficiencies in terms of assembling of the metagenomics data [36, 37]. Conversely, such techniques are usually designed for the production of single genomes. However, they do not work in close collaboration in situations when multiple species are present as shown in the environmental samples. One method is to deal with such complexities is to overcome the assembly and directly proceed to the finding genes.

New techniques are currently being implemented for the prediction of the genes within the metagenomics. Some of the new methods include MetaGene [38], Orphelia [39] and FragGeneScan [40.

2.3.1. MetaGene

MetaGene Utilizes the same approach as GeneMArk. hmm [41] that considers the GC-content sensitive dicodon and monocodon approaches that are computed completely

from the annotated genomes. As soon as MetaGene can extract all of the possible open reading frames (ORFs) within the fragments, it makes use of the statistical tools computed from completely annotated genomes to score the fragments.

The next stage carries out a dynamic programming algorithm which mixes the previous score with the length of the ORF, distance present within the ORF and its immediate neighbour as well as the distance found within the translation initiation start (TIS) and the left-most start codon. The dynamic programming algorithm aims to finalise the specific ORF sets while resolving the overlapping found within ORFs. The scoring criteria comprise of the log-odds ratios of the observed frequency within coding ORFs and the random ORFs. MetaGene uses two models; a specific model for archaea and the other one for bacteria. These models are selected automatically based on the results of the pre-defined classification of the domain technique during the process of classification. Randomly sampled fragments ranging in between 700bp attained from 12 annotated complete genomes are tested against MetaGene. The outcomes depict the capability of the MetaGene in terms of predicting the genes that have lower specificity and high sensitivity.

2.3.2. Orphelia

Orphelia offers better performance than MetaGene by making use of a two-stage machine learning technique. The first stage helps in building linear discriminants for the usage of the dicodon and monocode in addition to the TIS characteristics which are directly obtained from ORFs. During this stage, characteristics are extracted linearly from high-dimensional characteristics attained from the TIS data and codon usage, thereby decreasing the usage of every single characteristic. The next stage merges the characteristics attained from the length, GC content, and the linear discriminants through the use of a non-linear neural network that generates the possibility that the provided ORF can encode the protein. Lastly, Orphelia exhibits a post-processing algorithm that makes use of the probabilities from as the scoring technique to resolve the issues involving overlapping. Orphelia is also evaluated on the same principles as MetaGene yet many extensive instruments and experimentation have been carried out to analyze the impact of contrasting lengths of the fragments, program accuracy in terms of predicting the TIS and incomplete vs. complete prediction program ability.

2.3.3. FragGeneScan

Based on HMM (Hidden Markov Model), FragGeneScan is a type of algorithm that is capable to predict genes within metagenomics fragments and also incomplete genome [40]. This algorithm combines the sequence patterns used for start/stop codon, codon usage, and the error sequencing models utilizing HMMs. To select the most favourable path for hidden states, the Viterbi algorithm is utilised to create the observed nucleotide fragment. While comparing the accuracy of FragGeneScan against MetaGene for short reads, the comparable performance of both were achieved with no errors in sequence for simulated 700bp reads [40]. As opposed, for reads containing sequencing errors and short read, FragGeneScan demonstrated relatively better accuracy than MetaGene [40]. Then, Thomas K. Peuker in 1973 published the first description of TIN in 3D. He suggested a new method that decreases space and time (Peuker, 1969).

2.4. MGC: Metagenomic Gene Caller

On the basis of the two-stage machine learning technique, MGC [42] is similar to Orphelia program [39]. Models involving machine learning segregation learned from a disjoint set of partitions of a specific dataset performs more accurately than a single model which is gained from a complete dataset (Chan & Stolfo [43].

Separate models are learned by MGC for multiple pre-defined ranges of GC in comparison to the single model technique implied by Orphelia and apply the adequate model for every fragment on the basis of the GC content. Separation of the training information in MGC offered by GC-content ensures in the production of mutually exclusive data partitions that are required for training multiple models [42].

Partition is implied in the MGC method through GC-content for the training dataset. The relationship between the composition of the amino acid and the nucleotide bias-motivated in the utilisation of GC-content. Nucleotide bias has a drastic yet compelling influence on the composition of amino acid within the encoded protein (Singer & Hickey [44]. This impact is not only verified within entire genomes but is also relevant for individual genes/ Use of GC-content guarantees the separation within the model for different composition is regarded rather than collectively putting them in a single model [42].

GC-content has an impact on the usage of the codon that subsequently impacts the usage of amino acid. It was analysed that the utilisation of the GC-rich encoded amino acid codon enhanced by 1 per cent (approx.) for every 10 per cent rise in the GC-content genome (Lightfield et al.). In addition, this situation was also factual for the GC-poor codons. Separation of the GC-content into multiple ranges of GC could guarantee that multiple linear discriminants would be separated from the amino acid usage and codon in a highly précised manner [45].

2.4.1. MGC Algorithm

MCG, similar to Orphelia, is a two-step machine learning technique [39, 42]. The foremost step comprises of the linear discriminants which are utilised for compacting high dimensional characteristics space into small ones.

On the basis of the GC-content ranges, multiple linear determinants were provided with the training. Initially, the training information was segregated into different GC ranges that were defined to ensure that the training sequences within these ranges were similar. For instance, a GC spectrum was divided into ranges and every partition comprised of 10 per cent of the sequences of the training information (El Allali & Rose [42]. Later, the information acquired from every range was used to generate all the essential discriminants for computing the characteristic. The initial phase of MGC (See Fig 6) has demonstrated the linear determinant phase of the MGC for a specific GC range and has also illustrated all of the nine characteristics utilised in the 2nd phase of the MGC algorithm [109]. Fig. 5 has also represented the multiple ORF locations within a fragment.

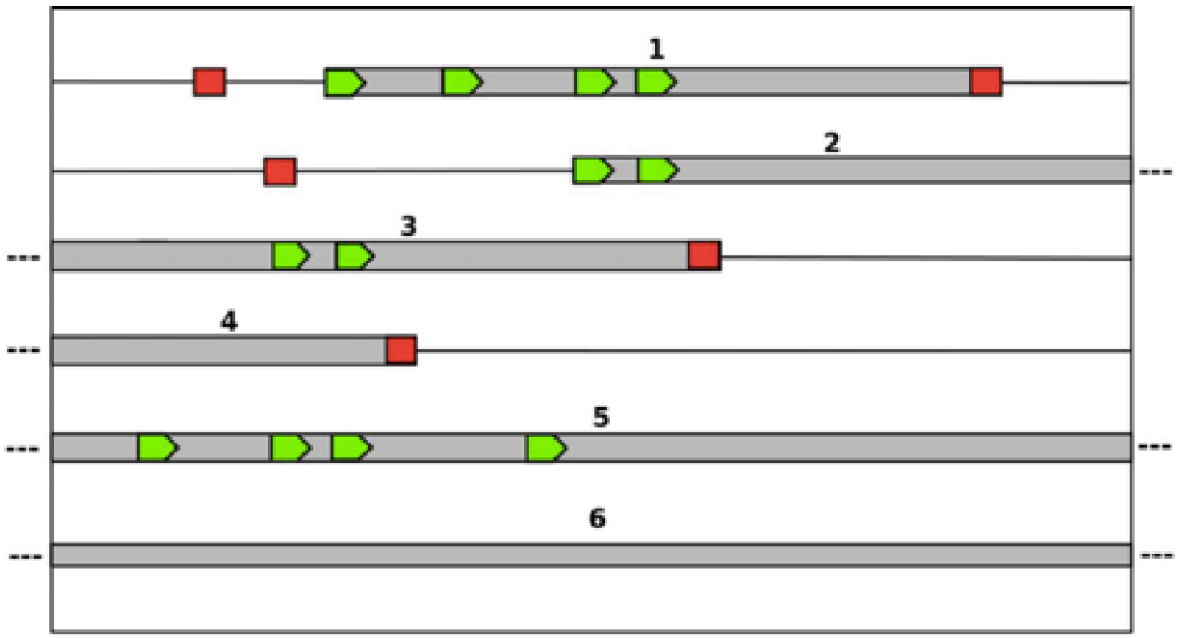

Figure 5. The possible ORF positions within the forward strand of a fragment. The fragment is depicted by the outside box and gray bars represent possible ORFs. Candidate translation initiation sites are represented by green pentagons and red squares indicate stop codons (Obtained from El Allali and Rose (42)

All of the possible incomplete and complete ORFs are extracted as soon as all the models are trained from the testing set similar to the process conducted during the training phase [109]. An equivalent neural network model (NN) is utilised for scoring ORF on the basis of the GC-content of the fragment. The resultant of the NN determines the estimation of the posterior possibility highlighting the coding of the ORF. In the second step (Refer to Fig. 6.2), the NN model is presented. Overlapping of the ORFs is resolved once all the hypothetical ORFs are scored. In a similar manner, Orphelia also uses a similar greedy algorithm to estimate the overlapping in between all the potential ORF candidates with a probability ratio higher than 0.5. Provided that the list of candidate for a specific fragment comprising of all ORFs with Probability more than 0.5, Algorithm 6 has discussed the selection scheme which is utilised for the creation of the final gene list [109]. Highest allowed overlapping (^{o}max = 60bp) that is the minimal length of the gene selected for the prediction

Table 2. The final candidate Selection Algorithm [42]

1: while L ≠ Ø do
2: Find i_{max} argmax$_i$P$_i$, $\forall i \in$L
3: Move ORF i_{max} from L to ł
4: Remove all the ORFs in L that overlap with ORF i_{max} By more than o_{max}
5: end while

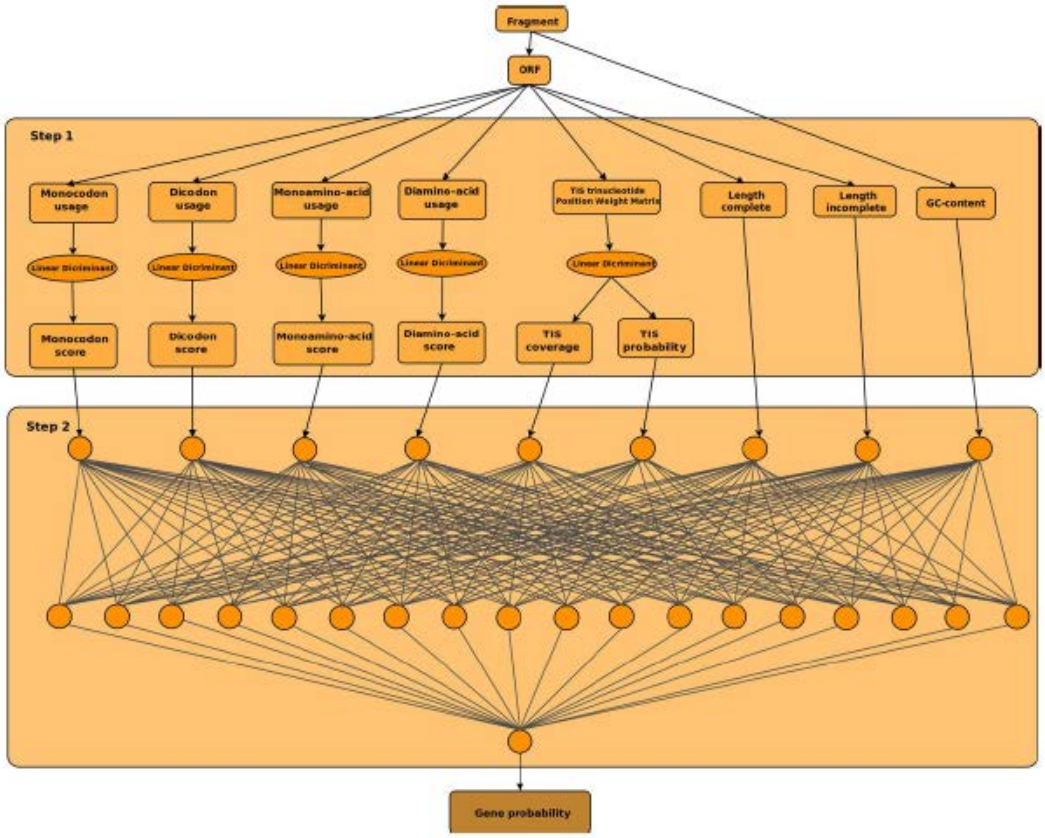

Figure 6: MGC's scoring scheme: The first steps computes six features from the ORF based on the corresponding linear discriminant and two additional features are computed directly from the ORF. The last feature is derived from directly the fragment. The neural network model from the corresponding GC-range is used to combine features from the previous step in order to compute a final gene probability. (Obtained from El Allali and Rose [42]

2.5. High-Performance Computing

One of the major challenges that designers of integrated circuits face is energy consumption. İt is not only demanding to provide power to the chip, but also there is a possibility of the power-driven heat causing major malfunctions. Attaining maximum density for scaling of the chips is eventually restricted by the system inability to cool down the circuit. Consequently, designing of the microprocessors within the semiconductor industry is projected on 2 trajectories. The many-core technique has offered more attention for executing the throughput required for corresponding applications. As opposed, the idea of the multi-core sustains the execution speed of the sequential programs when utilizing multiple cores. Many-core architecture is divided into a huge number of small cores. For instance, each cored within the NVIDIA GPUs is identified as heavily multi-threaded, in-order, single-instruction issue processor which shares a similar cache with CUDA since it is the GPU programming environment of GPU of NVIDIA.

The programming model of CUDA contains of the host as well as the device attributes. A highly specified device function which is known as Kernel function operates on GPUs to improve the computationally difficult and highly parallel procedures.

In conventional software applications, a large number of program segments chiefly include a huge amount of parallelism data, a property which promotes multiple arithmetic operations to be performed safely on different data structure programs simultaneously.

Since the recent GPUs are based on the single-instruction multiple data (SIMD) approach

2.5.1. Architecture of GPU

2.5.1.1. Streaming Multiprocessor

New streaming multiprocessor (SMX) by NVIDIA has proposed numerous architectural innovations. A single SMX comprises of 192 single-precision CUDA cores, 32 special function units (SFU), 64 double-precision units, and 32 store/load units. The count within the SMX ranges from 7 to 15 based on different chipsets.

Multiple cores of CPU are launched to work by the Hyper-Q within a single GPU that radically improves the percentage of the temporal occupancy upon the GPU. GPU’s

total number of connection with the host is also improved by the Hyper-Q that promotes 32 simultaneous processes.

GPU is enabled by the GPU Director which is a capability that allows GPUs within a single computer or different nodes in a network to exchange the data directly without the assistance of an external memory system. Access to memory from different GPUs is promoted by RMDA feature present in GPU Direct which gives access to third-party devices within numerous GPUs found in a single system to predominantly reduce the MPI send latency while receiving messages from/to GPU memory. The demand is also decreased upon the system memory bandwidth thereby freeing the GPU DMU-engines to be utilized by other tasks perform by CUDA.

2.5.1.2. Memory Model of GPU

There is slight variation within the GPU memory model as compared to previous versions. There is an additional cache memory that is denoted for read-only data.

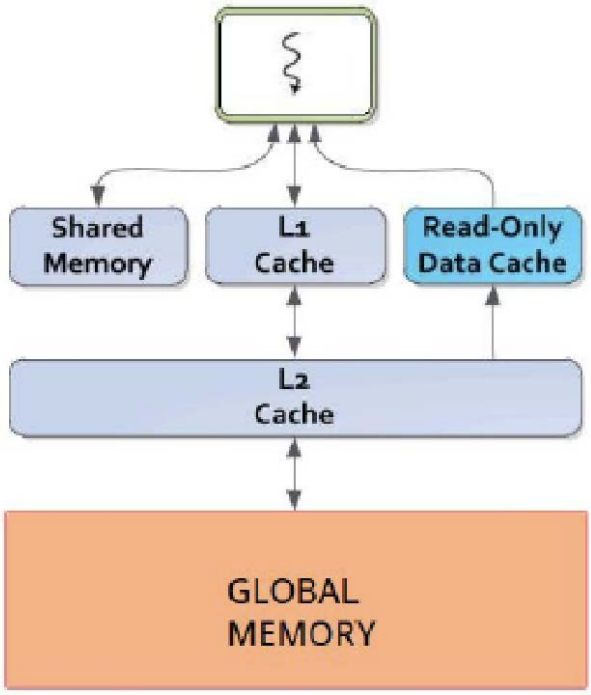

Figure 7. Memory hierarchy of a GPU thread (8)

2.6. Programming of GPU

2.6.1.CUDA

Compute Unified Device Architecture (CUDA) is defined as a parallel computing architecture that is produced by NVIDIA and initially available in 2007 [18]. The code of CUDA is executed on the device (GPU) or on the host (CPU). The code of CUDA on the host is dedicated for the initialization of the GPU, transferring data from/to GPU, and calling kernels (functions) within the GPU.

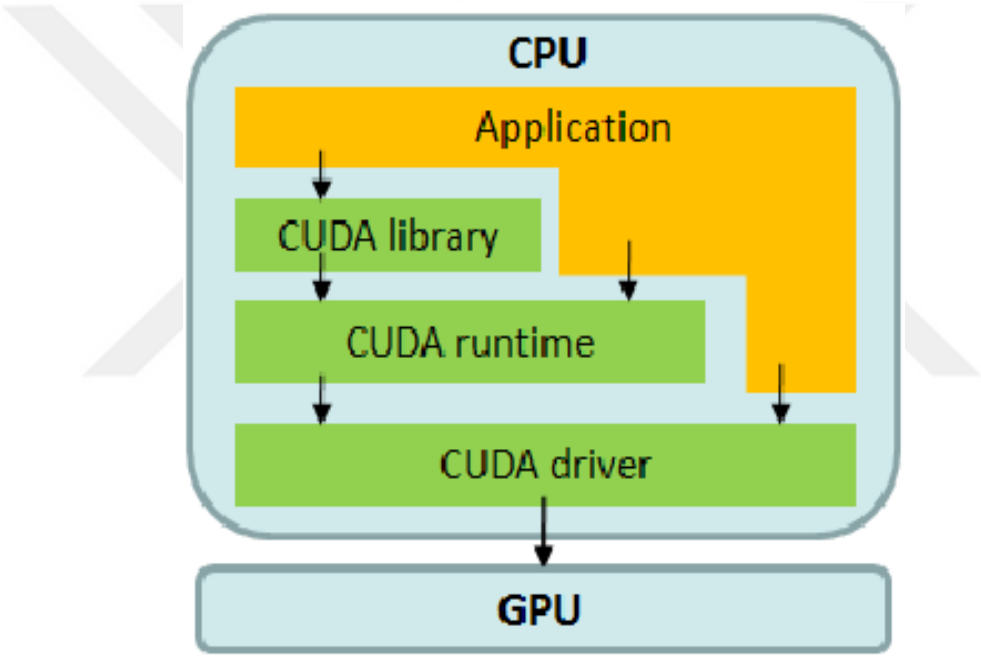

Figure 8. Overview of CUDA drivers, libraries, and kernels.

There are two forms of CUDA host code: CUDA Driver API and the C-for-CUDA. CUDA Driver API is regarded as a lower level CUDA version that is handle-based. On the other hand, C-for-CUDA is more like a C-programming language that enables to write programs for device controlling. Although, many objects are referenced through handles which could be passed to different functions for manipulating the objects in CUDA. However, for this project, C-for-CUDA was used. The kernel code of CUBA was assembled by NVIDIA's NVCC (NVIDIA Compiler for CUDA) into different *cubin* object code which was then transferred to the targeted GPU for final execution. The

functions of the Kernel were identified by the host code through the standard C-convention.

2.6.2. CUDA Matrix Addition Example

The following example has been discussed to demonstrate the comparison of a parallelised GPU-enabled program and the programming styles depicted by a tradition C-program. The example has also analysed the calculation highlighting the sum of both the matrices.

A code segment to be executed upon the PC written in C is illustrated in Fig 3.6a. The C-for-CUDA code which is also going to be executed on the PC is shown in Fig 3.6b. Calls are made by C-for-CUDA to GPU in order to execute a kernel that enables the calculations of the matrix (see Fig. 3.6c). The tradition code of C (Fig.3.6a) comprises of a nested for next loop restating single index at one time to estimate the sum of the matrix. Fig. 3.6b demonstrates the C-for-CUDA code which is running on the PC (host). This produces a matrix in addition to the GPU estimation by making a call to Kernel (Fig. 3.6c). A thread block of 16x16 dimensions is analysed that contains 256 threads. Keeping in mind that there is a limitation of 512 threads for every thread-block, one thread per matrix is mapped. A grid is also introduced to cover the MxN matrix estimation span within the thread-blocks (Fig. 3.7). All 16 multiprocessors in the GPU task are executed for the thread-blocks within the grid to enable the addition of the matrix

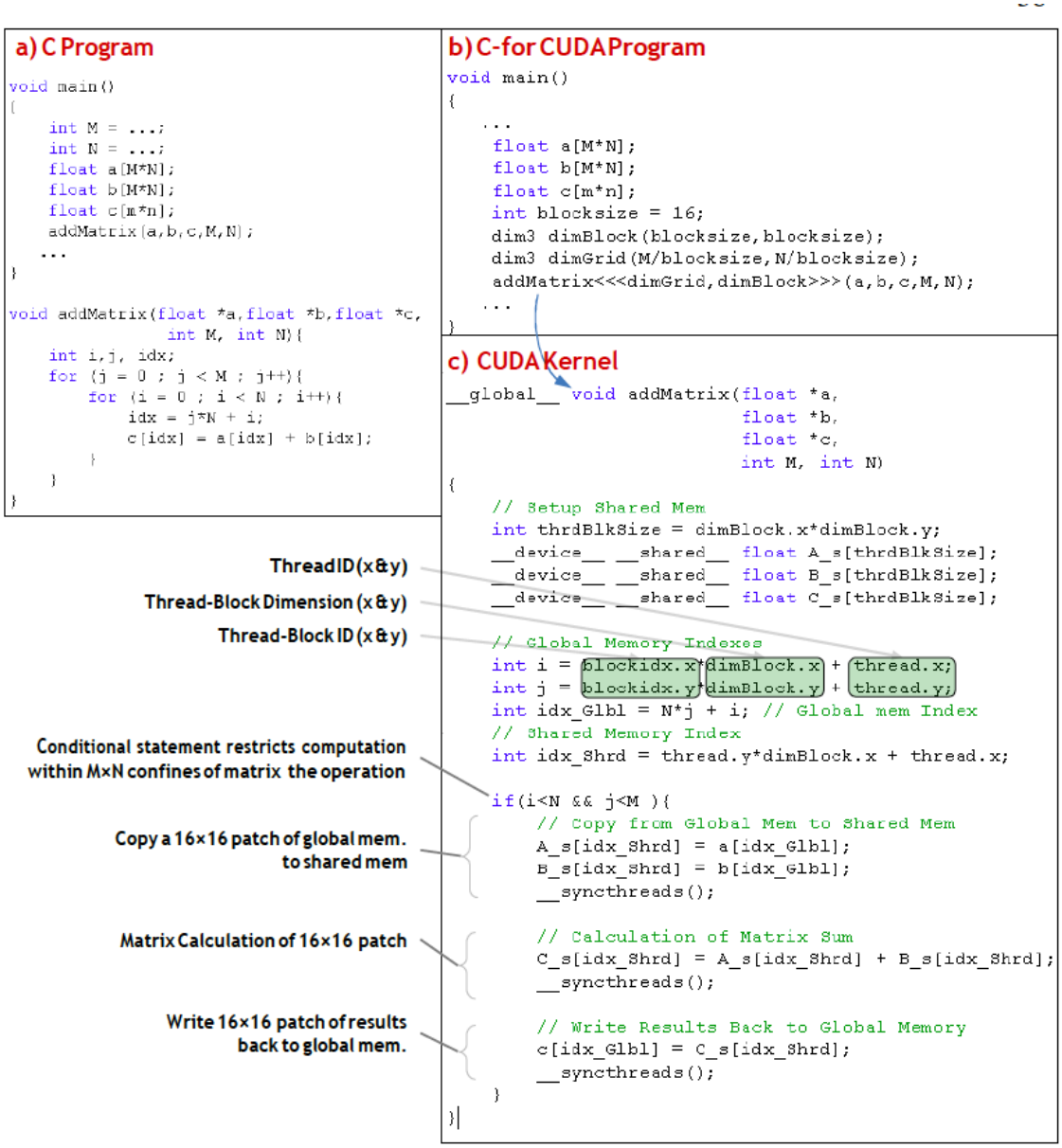

Figure 9. Programming sample of the matrix operation (C=A+B). (a) sample nested loop algorithm done in C.(b) The same matrix operation implemented in C-for-CUDA which launches (c) the kernel code on the GPU.

It is not unlikely to discover some overlapping in the thread-blocks which has been established to cover the MxN matrix of the estimation span. This is generally true when the size of the thread-block is fixed for compilation time yet the matrix dimensions are only specified during the run time and hence are user-defined. Fig 3.6c shows a conditional statement enlisted within the CUDA kernel to ensure that the thread-blocks which straddle the execution matrix bounds are only responsible for executing a single threads which fall within the calculation space. This restricts the kernel to MÅ~N dimension for every matrix

hence every matrix dimension is arbitrary and confined by the number of global memory present on the GPU.

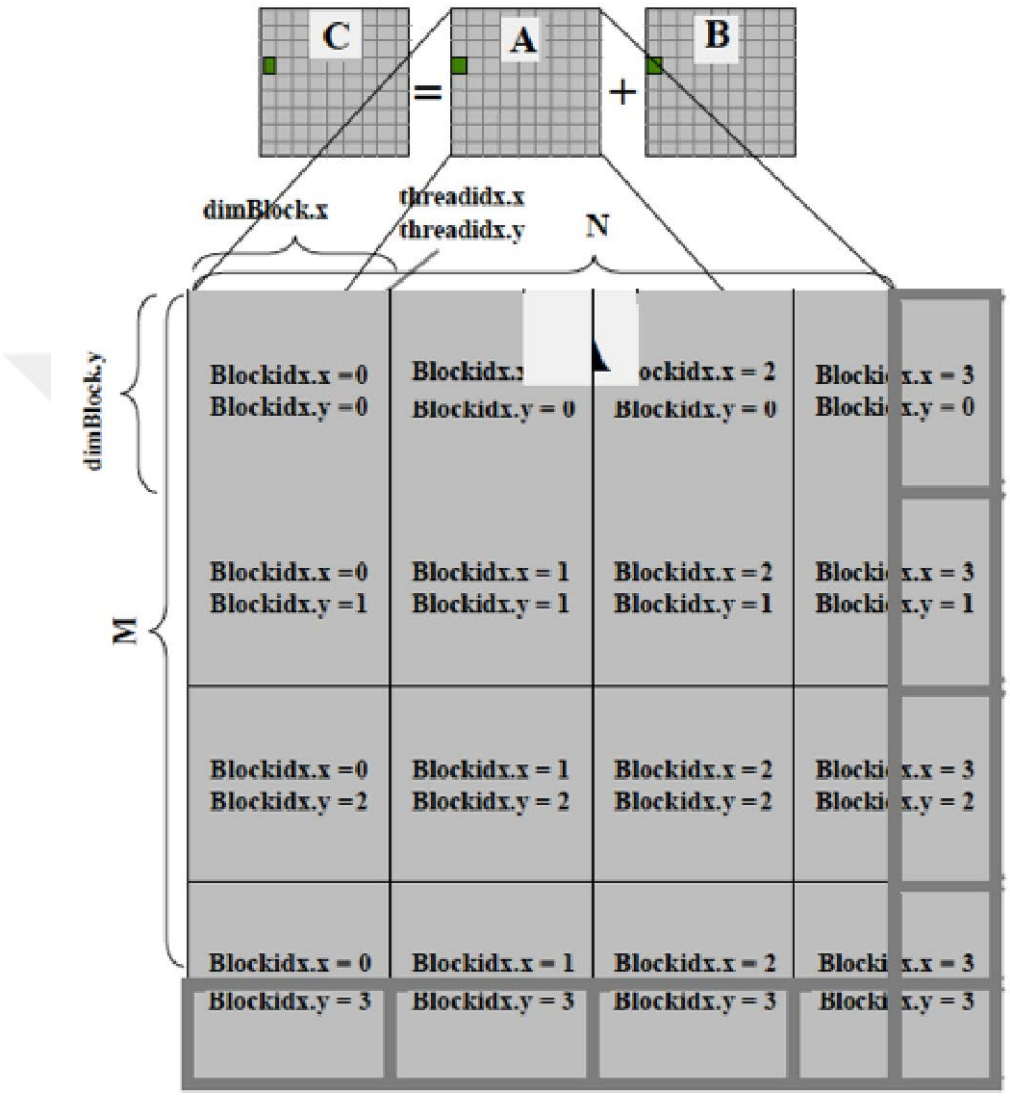

Figure 10. Grid layout of an example of a CUDA implemented access method to an M×N matrix using multiple thread-blocks.

Every code on the kernel within every thread is executed as follows. When multiprocessor executes a kernel, thread-block co-ordinates within the grid and the thread co-ordinates found in the thread-block goes through every active thread. As illustrated in Fig. 3.6c and 3.7, 'blockIdx.x' and 'blockIdx.y' represents the grid co-ordinate for a single thread-block. 'threadIdx.y' and 'threadIdx.x denotes the thread co-ordinates. Every active thread utilises only these coordinates to estimate the location within the computation space

where the thread is needed to be executed in the kernel code. Based on the location, every thread is given a task related to accessing a specified area within the global memory and replicates a 16x16 patch of matrics (A & B) to the shared memory of the multiprocessor. The matrix sum of the 16x16 patch within the shared memory has then computed the thread. The outcomes of the patch are stored on global memory. After all threads within the thread-block have been executed, the next set of the grid is provided to the multiprocessor and thread co-ordinates in response to the next available thread-block within the grid.

2.6.3. OpenCL

OpenCL is a framework that writes the program for executing the operating system and heterogeneous platforms. The chief element of OpenCL is its open standard language used for parallel programming of the processors which is available in PCs, embedded/handheld devices, servers, and GPUs. Apple originally developed OpenCL however it is currently operated by Khronos Group [48] with the involvement of different industry top-leading organisations and institutions. This also includes AMD and NVIDIA which are two of the topmost manufacturers of GPU. Khronos Group is in close collaboration with many too hardware manufacturing companies including Intel, Motorola, Apple, and Toshiba to align their hardware products or contribute their expert opinion in the growth of the open standard [49]. OpenCL GPU code is almost similar to the code written within the CUDA-Driver-API.

Also, there is a virtual similarity in the kernel code syntax [49]. OpenCL and CUDA have high control and power over the GPU hardware.

.

3. DETERMINATION OF GENE IN DNA SEQUENCES

As shown in the experimental results presented in the MGC study, individual learning models for many predefined GC regions instead of a single model approach improve the performance of the neural network. The addition of amino acid utilization properties has also been observed to increase the ability to correctly classify the MGC method. MGC algorithm The gene detection approach clustering data according to GC content offers a new perspective on the problem for other artificial intelligence based gene detection algorithms.

A microorganism culture can be mapped and linked to the organism by sequencing technique. However, metagenomics tries to blend the genetic information of microorganisms taken directly from their habitats without being cultured.

New generation (high volume) sequencing technologies can read sequences that are many times more than conventional sequencers. However, in high-volume sequence data obtained in metagenome sequencing, the reading error rate is quite high and the sequence reads are shorter.

However, sequences from a metagenome are noisy, partial, and more importantly, from thousands of different organisms. Although some studies have been successful in mapping gene cultures of cultured micro-organisms [50,51], considering these difficulties in metagenomics, researchers have aimed at identifying the genes present in the metagenome separately rather than making gene maps.

Several methods have been developed using different methods to identify genes in the metagenomics field. The newly developed methods are MGC [42], GeneMark.hmm [53], MetaGene [38], Orphelia [39], FragGeneScan [40].

MetaGene derives monocodon and dicodon discriminants derived from genes with specific GC content similar to the method of GeneMark.hmm. After extracting all possible “open reading frames” (ORF) from MetaGene, the score of each ORF in the sequences is statistically calculated. In the next step, using the dynamic programming technique, secondary parameters such as ORF length and distance between ORFs are combined with the previous score. By examining the overlaps of ORFs in this way, the strongest among the ORFs are identified. MetaGene uses two models for bacteria and archaeobacteria.

Orphelia performs better than MetaGene by using a two-stage machine learning approach. In the first step, the TIS properties of the ORFs as well as the linear disimerimants derived from monocodons and dicodons are calculated.

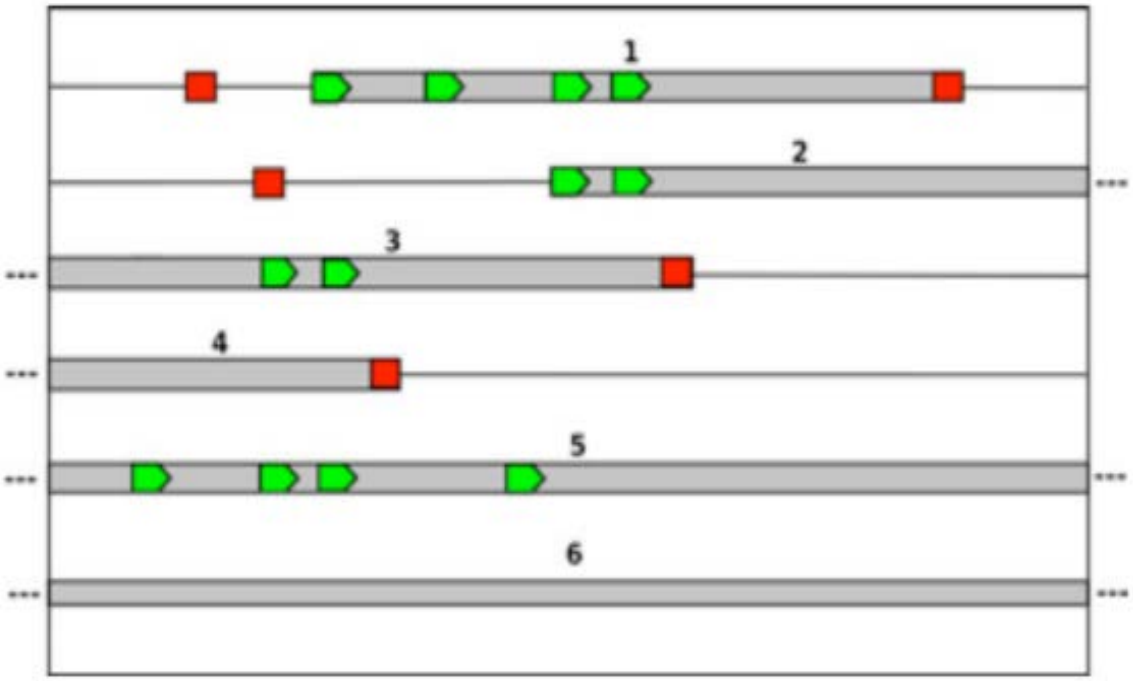

Figure 11.Determining possible ORF locations by scanning the sequences further (Source: Allali and Rose's MGC study [39]).

In the next step, Ophelia processes the data obtained in the first phase using a nonlinear artificial neural network to produce an estimated value to determine whether each ORF encodes protein.

FragGeneScan is an algorithm for gene extraction using hidden Markov models (HMM) [7]. This can detect both genes in all genomes and genes in metagenome fragments. Algorithm Using codon using HMM models the sequence order sequencing errors of start / end codons. The Viterbi algorithm processes the hidden states in the HMM according to the given nucleotide sequence and calculates the best path.

4. GENE EXTRACTİON ALGORİTHM FROM METAGENOME

The MGC algorithm, like Orphelia, identifies genes in metagenome sequences using a two-stage machine learning approach [39].

While Orphelia creates a single model taking into account all GC-content, MGC takes each GC-content as a model independent of the other, by breaking the GC-content into specific ranges. The development of the MGC was inspired by the work of Chan and Stolfo [43]. In this approach, dozens of artificial neural networks should be learned.

When we examine the application of MGC, we see that different artificial neural networks are used for each model instead of a simple artificial neural network with multiple outputs. Also, since there is an artificial neural network specific to MATLAB, each neuron has a as bias 'value.

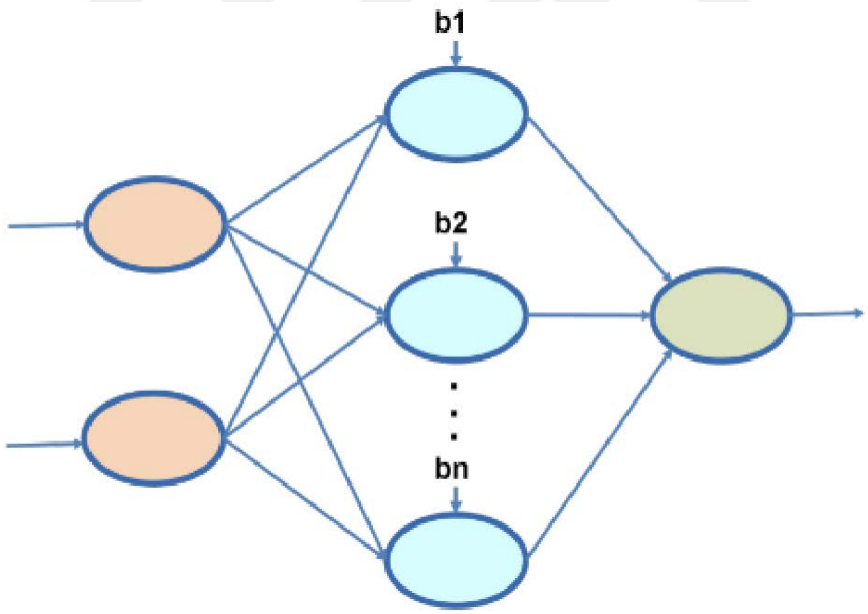

Figure 12. Operations are longer in Artificial neural networks (ANNS) with bias values

Figure 11 shows the gray marked regions of the candidate ORFs. The ORF start and end points are indicated by pentagons and red squares, respectively.

All possible ORFs in the data set are determined by linear scanning. Two different species of ORF were obtained. ORFs of the first type are referred to as whole because they have both the start codon (ATG, CTG, GTG or TTG) and the stop codon (TAG, TGA or TAA).

The set of fragment ORFs does not have either or both of the start or end codons. The reason for generating the fragment ORF cluster is because the gene is assumed to have its continuation in the other sequence. Allali and Rose take into account the ORFs of 60bp and longer in the fragmented ORF cluster in MGC application.

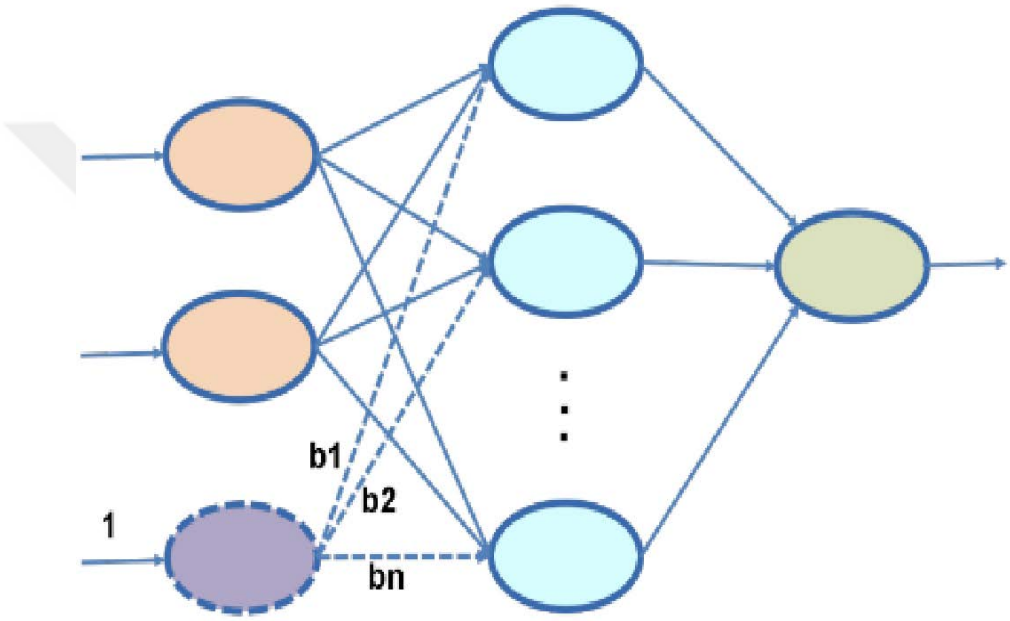

Figure 13. Deviation values can be simplified. A vector can be produced by multiplying the input matrix and output matrices

In MGC application, ANN was trained using some of the linear discriminates depending on GC - content ranges. Training data were defined so that the number of training sequences was the same in all these ranges and GC ranges were separated.

The method created in MGC can be examined in two parts. The first section shows the linear discriminant extraction step derived for a given GC range. In the second part, the structure of the ANN used for classification is given. The learning and classification stages of the MCG work in the same way.

In the first step, each ORF is pretreated and the value of the six linear discriminant properties is calculated. The last three characteristics are derived from ORF from the east.

The artificial neural network prepared in MGC consists of three layers. This artificial neural network is processed with 9 properties derived from ORFs. There is a 9 × 25 weight matrix and a 1 × 25 bias vector in the input layer of the artificial neural network. In the output layer, there is another 25 × 1 weight vector.

We can simplify an ANN with a deviation vector. For example, an artificial neural network with deviations in the intermediate layer, such as in figure 12, is given.

We can achieve a simpler network transformation in this ANN. Then we can reduce the input matrix whose constant weight values are multiplied by the output vector to a vector

The following example shows the deviation values of ANN. We can distribute these deviations by adding an artificial neuron. Figure 12 shows the addition of new neurons.

The input value of the newly added neuron should always be taken as +1. We can simplify the final ANN into a vector. We multiply the matrices of 3 X N and N X 1 to obtain a vector of 3 X 1. With this simplification process, we need an ANN calculation for each ORF, avoiding a total of millions or even billions of matrix multiplications. Instead, we make two vector products. Let's simplify the improvement by comparing the number of steps.

Equation 1 is used to calculate the number of operations required in the original data structure:

$$^{A}mat = \text{Input Layer} \times \text{Interlayer} + \text{Bias} + \text{Interlayer} \times \text{Output layer} \quad (1)$$

$$^{A}mat = 9 \times 25 + 25 + 25 = 275$$

If we adjust the deviation values of the neurons in the intermediate layer and multiply the matrices to obtain a vector, the number of operations decreases considerably. When the method is simplified, see equation 2 for the number of steps required:

$$^{A}vector = \text{Input vector} \times \text{Artıficial neural network(ANN) vector} \quad (2)$$

$$^{A}vector = (9 + 1) \times (9 + 1) = 100$$

In addition, the ability to create 1024 threads in one block of the GPU (our GPU supports this) allows multiple matrix multiplications at the same time.

In this study, we have observed that MATLAB programming platform is inadequate for applications requiring high computation with classical coding method. Of course, modules using GPUs have been developed in MATLAB. However, when we used MGC, this module was not available in the MATLAB version on the Stampede Supercomputer.

5. RESULTS AND FUTURE WORK

5.1. Results

As mentioned earlier, we developed the GPU kernel function for the whole part of the ANN only, not the entire MGC application. The test results of the developed GPU application are summarized in Table I. Two files containing 512000 and 1024000 ORFs were tested for three different block sizes for CUDA as indicated in the table.

Table 3. Performance shows GTX 755m graphics card for 512000 and 1024000 ORFs

ORF number	**Thread Count/Block**	$T_{sequence\ sec)}$	$T_{CUDA(sec)}$	$T_{OpenCL(sec)}$	$T_{OpenCLOpt(sec)}$
512K	256	17.452	0.891	3.242	1.330
	512	17.473	0.902		
	1024	17.496	0.906		
1024K	256	34.735	1.678	6.089	2.531
	512	35.209	1.703		
	1024	35.251	1.725		

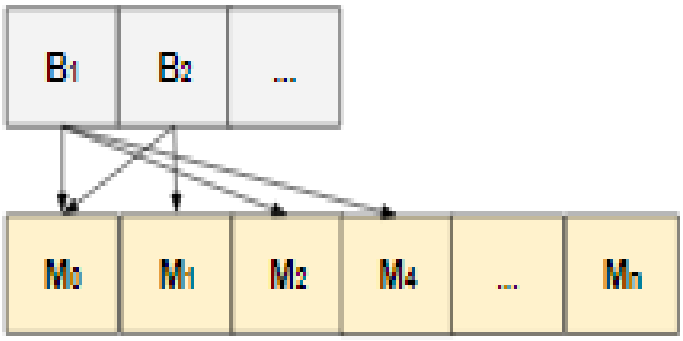

Figure 14. Distributed memory access (non-coalesced).

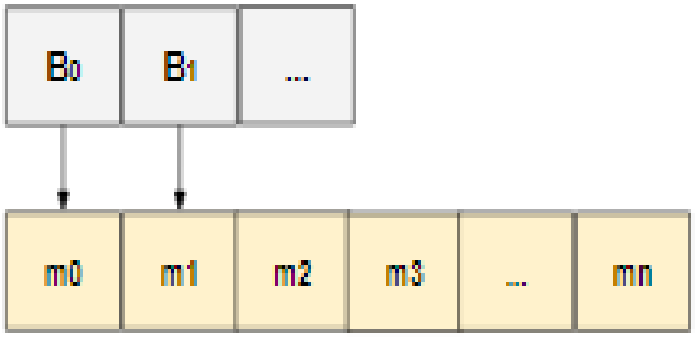

Figure 15. Cumulative memory access (coalesced)

In the table, the ^{T}S*equence* column indicates the time required for the serial running part of the application. The serial section of the application is equal to the sum of the time it takes to read the file containing the ORF information, transfer it to the GPU, and take the results to write to the file.

The ^{T}*CUDA* column indicates the runtime of the CUDA kernel function in the GPU. The last two columns are the run times of the kernel functions prepared in OpenCL. The TOpenCL column is the run time of the core function prepared in an optimized OpenCL environment. In kernel functions where the data is unregulated, thread blocks have irregular access to different parts of the memory (see Figure 14).

By regulating the memory access of vector processors, unnecessary waiting is eliminated. The illustration of this is shown in figure 15.

5.2. Future Work

In the future, we will transfer the entire metagenomic gene caller (MGC) application to the GPU platform. In this way, we will be able to produce a version of MGC that can classify faster and compare the results.

6. REFERENCES

1. Ward D.M.,Weller R., Bateson M.M. 16S rRNA sequences reveal numerous uncultured microorganisms in a natural community. Nature, 345:63–65, 1990.
2. Goebel U.B. Phylogenetic amplification for the detection of uncultured bacteria and the analysis of complex microbiota. Journal of Microbiological Methods, 23:117–128, 1995.
3. Gonzalez J.M., Saiz-Jimenez C. Application of molecular nucleic acid based techniques for the study of microbial communities in monuments. Intl Microbiol, 8:189–194, 2005.
4. Chen K., Pachter L., Bioinformatics for whole-genome shotgun
5. Toward precision medicine building a knowledge network for biomedical research and a new taxonomy of disease National Research Council U.S. Committee on A Framework for Developing a New Taxonomy of Disease. 2011.
6. Trajanoski Zlatko, *Computational Medicine* IX, 203p, 2012.
7. Wetterstrand K.A., DNA *sequencing costs*: Data from the NHGRI large-scale genome sequencing program [http://www.genome.gov/sequencingcosts] accessed Jan 1 2014.
8. Savran, Ibrahim. 2014., 'High-performance Meta-genomic Gene' PHD Thesis - University of South Carolina, USA
9. J. Cohen, 'Bioinformatics - an introduction for computer scientists' *ACM Computing Survey*s, vol. 36, no. 2, pp. 122{158, 2004. [Online]. Available: https://dl.acm.org/citation.cfm?doid=1031120.1031122
10. P. Enrique Reynaud, 'Protein misfolding and degenerative diseases,' *Nature Education*, vol. 3, no. 9, p. 28, 2010. [Online]. Available: http://www.nature.com/scitable/topicpage/protein-misfolding-and-egenerativediseases- 14434929.
11. 'What are proteins and what do they do?' https://ghr.nlm.nih.gov/primer/ how genes work/protein, October 2017.
12. 'Specificity of enzymes' http://www.worthington-biochem.com/introbiochem/ specificity.html.
13. L. Stryer and J. M. Berg, *Biochemistry 5th edition*. New York: W H Freeman, 2002.[Online].Available: https://www.ncbi.nlm.nih.gov/books/NBK22380/
14. A. Purcell, 'Dna,' https://basicbiology.net/micro/genetics/dna/, February 2016.
15. B. Cornell, 'Dna sturcture,' http://ib.bioninja.com.au/standard-level/topic-2-molecular-biology/26-structure-of-dna-and-rna/dna-structure.html
16. R. Kaur, 'Dna replication,' https://ramneetkaur.com/dna-replication/, August 2016.
17. A. Sugino, S. Hirose, and R. Okazaki, 'Rna-linked nascent dna fragments in escherichia coli,' *Proceedings of the National Academy of Sciences*,vol. 69, no. 7, pp.1863{1867, 1972. [Online]. Available: http://www.pnas.org/content/69/7/1863
18. B. Cummings, 'Dna replication' http://academic.pgcc.edu/_kroberts/Lecture/

Chapter%207/replication.html.
19. 'The genetic code'" http://www.biology-pages.info/C/Codons.html, December 2016.
20. 'translation / rna translation,' http://www.nature.com/scitable/definition/translation-173, 2014.
21. S. Clancy and W. Brown, 'Translation: Dna to mrna to protein," https://www.nature.com/scitable/topicpage/translation-dna-to-mrna-to-protein-393, p. 101, 2008.
22. M. Vötsch, 'mrna degradation' http://www.eb.tuebingen.mpg.de/research/research-groups/remco-sprangers/mrna-degradation.html.
23. 'Protein structure,' http://www.nature.com/scitable/topicpage/proteinstructure-14122136, 2014.
24. E. Arunan, G. R. Desiraju, and R. A. Klein, 'Definition of the hydrogen bond (iupac recommendations 2011),' *The Scientific Journal* of IUPAC, vol. 83, July 2011.
25. Ionic and covalent bonds,' https://chem.libretexts.org/Core/Organic Chemistry/Fundamentals/Ionic and Covalent Bonds.
26. D. Goodsell, 'Chaperones,' https://canvas.instructure.com/courses/1021937/pages/levels-of-protein-structure, August 2002.
27. Sanger F., Coulson A.R. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. J. Mol. Biol. 94/3:441–8, 1975.
28. Sanger F., Nicklen S., Coulson A.R., DNA sequencing with chain-terminating inhibitors. Proc. Natl. Acad. Sci. U.S.A. 74/12:5463-7, 1977
29. Sanger F., Air G. M., Barrell B. G., Brown N. L., Coulson A. R., Fiddes J. C., Hutchison C. A., Slocombe P. M. et al. Nucleotide sequence of bacteriophage _X174 DNA. Nature, 265/5596: 687–95, 1977.
30. Adams M.D. et al., The genome sequence of Drosophila melanogaster. Science, 24/287(5461):2185-95, 2000.
31. Venter J.C. et al. The sequence of the human genome. Science, 16/291(5507):1304-51,2001.
32. Venter J.C. et al. Environmental genome shotgun sequencing of the Sargasso Sea. Science, 2/304(5667):66–74, 2004.
33. Ten Bosch J.R., Grody W.W., Keeping Up with the Next Generation. The Journal of Molecular Diagnostics, 10(6):484–492, 2008.
34. Tucker T., Marra M., Friedman J.M., Massively Parallel Sequencing: The Next Big Thing in Genetic Medicine. The American Journal of Human Genetics, 85(2):142–154, 2009.
35. Lin Liu, Yinhu Li, Siliang Li, Ni Hu, Yimin He, Ray Pong, Danni Lin, Lihua Lu, Maggie Law, Comparison of Next-Generation Sequencing Systems. Journal of Biomedicine and Biotechnology, 1–11, 2012.
36. Chaisson MJ, Pevzner PA: Short read fragment assembly of bacterial genomes. Genome Research 2008, 18(2):324-330.
37. Butler J, MacCallum I, Kleber M, Shlyakhter IA, Belmonte MK, Lander ES, Nusbaum C, Jaffe DB: ALLPATHS: De novo assembly of whole-genome shotgun microreads. Genome Research 2008, 18(5):810-820 [http://www. ncbi.nlm.nih.gov/pubmed/18340039].
38. Noguchi H, Park J, Takagi T: MetaGene: prokaryotic gene finding from environmental genome shotgun sequences. Nucleic Acids Research 2006, 34(19):5623-5630 [http://www.ncbi.nlm.nih.gov/pubmed/17028096].

39. Hoff KJ, Lingner T, Meinicke P, Tech M: Orphelia: predicting genes in metagenomic sequencing reads. Nucleic acids research 2009, 37(Web Server):W101-5 [http://www.ncbi.nlm.nih.gov/pubmed/19429689].
40. Rho M, Tang H, Ye Y: FragGeneScan: predicting genes in short and error prone reads. Nucleic Acids Research 2010, 38(20): e191.
41. Borodovsky M, Mills R, Besemer J, Lomsadze A: Prokaryotic gene prediction using GeneMark and GeneMark.hmm. Current protocols in bioinformatics editoral board Andreas D Baxevanis et al 2003 [http://www. ncbi.nlm.nih.gov/pubmed/18428700], Chapter 4: Unit4.5.
42. El Allali Achraf, Rose John R., MGC: a metagenomic gene caller. BMC Bioinformatics, 14/9: S6, 2013.
43. Chan P.K., Stolfo S.J., A comparative evaluation of voting and meta-learning on partitioned data. Proc 12th International Conference on Machine LearningMorganKaufmann,90-98,1995. [http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.7713].
44. Singer G.A., Hickey D.A., Nucleotide bias causes a genomewide bias in the amino acid composition of proteins. Molecular Biology and Evolution,17(11):1581-1588,2000. [http://www.ncbi.nlm.nih.gov/pubmed/11070046].
45. Lightfield J., Fram N.R., Ely B., Across Bacterial Phyla, Distantly-Related Genomes with Similar Genomic GC Content Have Similar Patterns of Amino Acid Usage. PLoS ONE 6(3):12, 2011.
46. NVIDIA Corporation, NVIDIA CUDA Programming Guide, June, 2011.
47. AMD Corporation, ATI Stream Computing OpenCL Programming Guide, Aug. 2010.
48. NVIDIA CUDA: Compute Device Architecture Programming Guide, v3.0, NVIDIA Corporation, Santa Clara, CA, USA.2010
49. "Khronos Opencl API Registry."The Khronos Group: Open Standars , Royalty Free,DynamicMediaTechnologies.Internet: https://www.khronos.org/registry/OpenCL/ [May 2010]
50. Chaisson M.J, Pevzner P.A. "Short read fragment assembly of bacterial genomes." Genome Research. 2008;18(2):324-330.
51. Butler J., MacCallum I., Kleber M., Shlyakhter I.A., Belmonte M.K., Lander E.S., Nusbaum C., Jaffe D.B. "ALLPATHS: De novo assembly of whole-genome shotgun microreads." Genome Research. 2008;18(5):810-820.
52. Borodovsky M., Mills R., Besemer J., Lomsadze A. "Prokaryotic gene prediction using GeneMark and GeneMark.hmm." Current protocols in bioinformatics editoral board Andreas D Baxevanis et al. 2003.

.

CURRICULUM VITAE

Shafiei Abdi Suleiman was born in Buhodle Somalia in November 1989. He graduated from Ilays Secondary School, Lasanod, Somalia in 2009. He got his B.Sc. from the computer science department at Nugaal University, Lasanod, Somalia in 2014. He has a good level of Turkish, Arabic, and English languages.