NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

PanCGP: Pangenome and Comparative Genome analysis Pipeline

by

Muhammad Ahsan

A thesis submitted in partial fulfillment for the degree of Master of Science

in the
Computational Science & Engineering
Research Center for Modeling & Simulation (RCMS)

April 2016

Declaration of Authorship

I, Muhammad Ahsan, declare that the work embodied in this dissertation is the result of work produced by me and has not been submitted for a higher degree to any other institution.

Signed: ______________________________

Date: ______________________________

"Learning is the process whereby knowledge is created through the transformation of experience."

David Kolb

Abstract

Since late 1990s, the development in the field of sequencing techniques has fasten the pace of gradually evolving field of genomics. In 1995, for the first time, a living organism genome was sequenced. It realized the fact that by examining the overall potential of a genome of an organism, the results can extend the possiblilities to the identification of gene or a mixture of genes. Along with advacement in computing technologies, this idea results in the developement of advanced next generation computational sequencing techniques and made possible the genome/proteome sequences of commercially significant organisms and pathogens in an optmized time frame and minimal cost. In this thesis, we present the realization of the implementation of Pan-genome and Comparative Genome analysis Pipeline tool using high performance parallel computing techniques. The aim of developing this high performance and scalable pipeline is to reduce the time cost of calculating the pan-genomes from the given dataset of protein sequences of bacterial strains. The pipeline is able to compute pan-genome analysis from unpublished datasets as well. Pan-genome and Comparative Genome analysis Pipeline (PanCGP) use divide and conquer approach to parallelize the whole analysis process. Data decomposition technique is used to break down the problem into smaller chunks and process them separately over the available processing resources. Since, each data chunk is being processed separately, this technique drives out the communication overhead during parallel processing of the pipeline and makes it embarrassingly parallel. PanCGP is able to scale on shared memory architectures and distributed memory architectures as well as hybrid architectures seamlessly. Scalability of pipeline depends on the size of input dataset as well as the number of available computing resources. MPJ Express has been used to exploit the pipeline on shared as well as distributed memory architectures in a seamless manner. Modular nature of PanCGP makes it highly customizable and ease of extending it to add more functional modules to the pipeline. Dataset of 38 strains of *helicobacter pylori* has

been given as input to PanCGP and CMG-biotools pan-genome analysis pipeline. The resultant time cost of PanCGP is much less ($\sim$ one sixth) compared to that of CMG-biotools benchmarks. Incorporating the same tools and versions of tools in PanCGP pipeline, best result accuracy has been achieved. The accuracy of results may vary depending on the versions of tools being incorporated. The software package can be downloaded from https://github.com/TechnologyCell/PanCGP and https://sourceforge.net/projects/pancgp/.

Aknowledgements

All the glories, praises and Love are for Almighty ALLAH, the Most Merciful, the Most Gracious, Who bestowed me the thinking power, ability and potential for donating a diminutive scientific knowledge, wet eyes and shuddering lips praise for HIS beloved The Holy Prophet Muhammad(P.B.U.H) for enlightening my integrity with the quintessence of faith in Allah, congregating all His kindness and mercy upon him.

In person I would like to prompt my inner, profound and sincere gratitude to my supervisor, Dr. Jamil Ahmad and Dr. Amjad, I will always do reverence and reminisce to their fatherly behavior, sympathetic attitude, perceptive pursuit, sentient direction, rational supervision, cheering perception and the most that I like their scholarly criticism. I extend deep emotions of appreciation, expressing my gratitude and indebtedness to Sir Tariq Saeed, Assistant Professor, RCMS, NUST, Sir Rehan Zafar, Assistant Professor, RCMS, NUST and Madam Humaira Gul, Lecturer, Research Center for Modeling and Simulation, NUST for their valuable guidance. Sir Tariq Saeed's strong experience in field of High Performance Computing and step by step suggestions and guidance throughout the work let me complete this project.

Words are lacking to express my modest obligation to my affectionate Grand Father Ch. Inayat Ullah jovindha(late) and other family members for their adoration, worthy wishes, faith, inspirations, incessant prayers for me and monetarily help without which the present ambitious would have been mere a dream. What are I'm just because of them.

If there were reveries to sell, merry and sad to tell and crier rings the bell, what would you buy, I will say very first moment that "University fascinating days". I know it's intolerable but it shows my blind love to university life. I am particular to thanks my all other friends especially Muhammad rizwan, Raheel Khan, Ehsaan Adeeb, Saqib Khan, Syed Sajjad Sattar, Arslan Ahmad, my fellows and seniors.

Finally, I make an apology for those; I have hurt or offended because the inaccuracies remain mine. May Allah bless all these people with protracted, exultant and peaceful lives. (Ameen)

Contents

List of Figures

List of Tables

Dedicated to My Grand Father Ch. Inayat Ullah Jovindha(late), Family and Friends.

Chapter 1

Introduction

In public gene databases like GenBank [1] and GOLD (Genomes OnLine Database) [2], there is a gradual increase in the discovery of sequenced genomes. According to National Centre for Biotechnology Information, in 2000, there were only 42 genomic sequences present in the list of NCBI. Since then, list has grown to 16108 genomes in April 2016[1]. This encouraged the development of tools and techniques for gene comparison of different species. Such comparisons include the identification of gene set common in all isolates of a group of interest (core genome) as well as dispensable/strain-specific genes. A known approach for identifying homologous sequences is reciprocal BLAST and various software packages are available for this task. By combining these tools in an optimized manner with the growing number of sequenced genes, past studies enlightens the fact that bacteria are actually the mixture/mosaics of different kind of genes including those common to all isolates as well as unique/strain-specific and partially shared among different strains of a specie [3].

1.1 Pan-genome analysis

Pan-genome is a new way of visualizing genetic information of the living organisms on Earth. This concept first used by Tettelin et al. [4] when they described *Streptococcus agalactiae* as the core-gene set among the six sequenced strains. In microbiology, Pan-genome [5] is full complement of genes in a specie. It is a union

[1]http://www.ncbi.nlm.nih.gov/genome/browse/

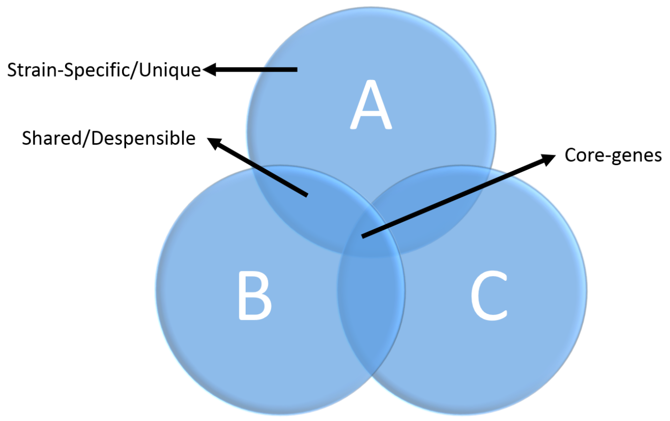

Figure 1.1: Pan-genome of three genomic strains A, B and C. *Core genes* are genes present in all three genes. *Shared/Dispensable genes* are those which are common to more than one strain. *Unique genes* are specific to every strain and not present in any other strain.

of all gene-sets of all strains of a specie as shown in figure 1.1. Pan-genome is classified into three main categories:

1. Core genes
2. Shared/dispensable genomes
3. Unique/Strain-specific genes

Core genes are common among all strains of a specie. They can be described as an intersection of gene-sets of all strains in a specie. *Shared/Dispensable genomes* shared by two or more strains of a specie. *Unique/Strain-specific genes* present in a single strain of a specie.

1.2 Computational Pan-genome analysis using High Performance Computing

Computational Pan-genome analysis is a complex computation problem that can be optimized using high performance computing techniques. This problem can be divided into four major sequential steps: Pre-processing, gene/protein comparison, clustering and report generation. Pre-processing includes fasta decomposition and hashing the sequences for their unique identification throughout the pipeline.

Gene/protein comparison step comprises of gene/protein database creation, a pre-step required for gene/protein comparison, followed by gene/protein comparison using reciprocal BLAST [6, 7], applying the fifty-fifty rule on BLAST [6, 7] output using TIGRCUT, a script which takes blast results in XML format and parses on those blast hits that satisfy to the 50-50 criteria. Protein/gene clustering is a process in which homologous sequences are grouped into related families based on similar properties. Results from gene/protein comparison is clustered based on similar properties. Report generation includes processes to extract core genes/proteins, unique/strain-specific genes/proteins, new gene/protein families and total number of genes/proteins found. Based on these computational results, a report is generated.

1.3 Reason/Justification for the Selection of the Topic

Advancements in gene/protein sequencing give rise to comparative genomics and related technologies. The increasing rate in the amount of gene/protein sequencing prompts the need of advance computational techniques for pan-genome extraction in an optimized timeframe. High Performance Computing generally refers to parallel/stream computing that gives a paradigm shift, the way advance and optimized algorithms were being made. Unlike conventional systems, it follows divide and conquer approach to improve work efficiency and efficacy. By applying HPC techniques in the field of comparative genomics, the problem is divided upto the fgranular level and is solved it in parallel fashion. The final results are aggregated which greatly fine tunes the performance as compared to other conventional pan-genomic analysis techniques as well as the other software packages available for that purpose.

1.4 Objectives

Comparative genomics enlightens the influence of genomics. It highlights the diversity of information that can be found even within a single species. PanCGP

(Pan-genome and Comparative Genome analysis Pipeline) tool is designed for microbiologist having minimum knowledge of computational analysis to gain maximum benefit in lesser time. The objective is to facilitate the users to perform computational pan-genomic analysis in a fraction of time with just a list of sequenced genomes/proteins as compared to other software packages already available for this purpose. It is designed to automatically scale depending on the available resources ranging from a personal computer to a hybrid cluster as well as the size of input. It also addresses the issue of large amount of data processing for comparative analysis of multiple genes/proteins.

1.5 Relevance to National Needs and Advantages

PanCGP (Pan-genome and Comparative Genome analysis Pipeline) tool expects to fasten the research procedures in the field of genomics by optimizing different time consuming steps pertaining to sequence analyses. This will ultimately shorten the duration of research, leading to less expenditure in terms of human resources, time and cost.

1.6 Areas of Application

Areas of applications of PanCGP (Pan-genome and Comparative Genome analysis Pipeline) tool are as follows:

- Discovery of new genes and gene family expansion
- Aid in group selection of comparative genomics
- Drug design
- Vaccine development

1.7 Composition of Thesis

The whole research work is structured as follows: Chapter one encompases the purpose and objectives of the research with its relevance to national needs and applications in different research fields. Chapter two covers a detailed literature survey in which existing pan-genome analysis tools and databases has been discussed. Steps involved in the pan-genome analysis has been elaborated. Importance of high performance computing and java language with its advantages and reasons to adopt it as a language for programming high performance computing languages and libraries used for the purpose.

In chapter three, there is a detailed discussion on methodology purposed for robust pan-genome analysis and introduction to PanCGP pipeline tool.

Chapter four describes the results obtained through PanCGP pipeline and its comparison with other tools. Chapter five describes the final conclusion and discuss some future work.

Chapter 2

Literature Review

Next generation genome sequencing technologies have resulted in a huge amount of sequenced genome data in sequenced genome databases. Fast rate of data aquisition for sequenced genome has flourished the field of genomics. Reduced cost of genome data acquisition methods has given rise to rapid genome data growth. In pan-genome analysis of bacterial species, the result majorly comprised of core and auxiliary genes [8]. It is an important origin of gene variation in the bacterial genome and enable the extension of the sub-population groups of bacteria to adapt to the specific sector in adjacent gene transfer. Low cost and high throughput genome sequence information platform has created dramatic increase in the species of bacteria and the opportunity to study their pan-genomes.

2.1 Existing tools

In the past decade, the significant progress in the development of DNA sequencing technology and its applications has led to a remarkable growth of genomic data. Especially in the case of prokaryotic genomes in which each of them extends to only a few mega-bases. In the coming decades, it is expected that more data will be collected as compared to the amount of data available now. Therefore, improvements in the existing databases and construction of new ones along with the development of fast and scalable tools, are especially require. This will engender the other related fields of genomics including pan-genomics and metagenomics, allowing them to build, share and quarry the incoming genomic data deluge. Since 2010, a dozens of software tools and packages have been developed, which are capable

Table 2.1: Software tools and packages already available for pan-genome analysis

Software	**Single Node**	**Multi-Node**	**Platform Supported**	**GUI**	**References**
CAMber	Yes	-	Linux	-	M. Wozniak et al., 2011 [9]
Panseq	Yes	-	Linux Windows	-	C. Laing et al., 2010 [10]
PGAT	-	-	Web	Yes	Brittnacher MJ et al., 2011 [11]
PanCGHWeb	-	-	Web	Yes	J.R. Bayjanov et al., 2010 [12]
PGAP	Yes	-	Linux	-	Zhao et al., 2012 [13]
ITEP	Yes	-	Linux	-	M.N. Benedict et al., 2014 [14]
Harvest	Yes	-	Linux	-	T.J. Treangen et al., 2014 [15]
GET_ HOMOLOGUES	Yes	-	Linux	-	Contreras-Moreira & Vinuesa, 2013 [16]
PanCake	Yes	-	Linux Windows	-	C. Ernst et al., 2013 [17]
PanGP	Yes	-	Linux Windows	Yes	Y. Zhao et al., 2014 [18]
PANNOTATOR	Yes	-	Web	Yes	A.R. Santos et al., 2013 [19]

of gene/protein sequence alignment and comparison, orthologous gene clustering, single nucleotide polymorphisms (SNPs) identification, construction of phylogenies and pan-genome analysis regarding multiple gene and protein sequences. Although they may share similar characteristics, each has its own limitations as well as features, making the room for future enhancement. Some tools and packages are listed in Table 2.1 and briefly introduced.

2.1.1 CAMber

CAMber [9] is a computational tool for comparative analysis of multiple isolates of bacterial strains to identify multigene families/clusters after refinement. To produce a refined and comprehensive annotation results, every gene in a multigene family ensures one-to-one correspondence with other gene, allowing to have integrated annotation results in the bacterial strains.

2.1.2 Panseq

Panseq [10] takes a multiple genomic sequences and user-defined parameters to calculate the core and auxiliary genes. It identifies regions unique to a gene or specfic gene group, classification of SNPs among dispensable genomic regions. It makes files on both the presence as well as absence of gene specific or shared regions and their usage in phylogeny programs, SNPs identification in core regions. The final report generated gives the graphical overview of the results obtained.

2.1.3 PGAT

A web-based tool originally designed as a platform to produce fast analysis of sequenced genomes of bacteria by next generation technologies. The **PGAT** [11] analysis tool comprises of three parts including a web interface as front end allowing user interaction to submit genome queries or user defined gene sets and pan-genome identification, a gene database and multi-genome annotation pipeline for bacterial genomes.

2.1.4 PanCGHWeb

PanCGHWeb [12] is a web-based tool designed for pan-genome analysis of microarrays using PanCGH algorithm. It facilitates genotype analysis as well as pan-genome microarray analysis as a cheap alternative to DNA sequencing. Comparative to standard CGH techniques, PanCGH assists in determining more accurate genomic content. The analysis results can be used to analyse complex hybridization data in a simplified way as well as an obvious way to realise the diversity among the related genomic strains.

2.1.5 PGAP

A highly efficient pipeline for analysis of pan-genome. **PGAP** [13] is able to perform five different types of analysis including functional gene analysis, profile analysis of pan-genome, evolutional analysis of species, gene cluster analysis based on functional enrichment and functional gene analysis based on genetic variation. The tool is Linux based and performance has been analysed using eleven strains of *Streptococcus pyogenes.*

2.1.6 ITEP

ITEP [14] is a flexible and powerful toolkit used for computational analysis related to curation and generation of protein families. Its modular design makes it highly customizable to future extension and optimization with the evolution of technology and software upgrades. By harnessing the power of comparative genomics and their integration with draft metabolic networks. It builds confident links between phenotype and genotype as well as high quality gene annotations to pave the way to evolution in metabolic network contexts.

2.1.7 Harvest

Harvest [15] is an interative platform for dynamically visualizing core-genome alignments. It is a collection for core genes alignment as well as provide visualization platform for studying behaviours including recombination detection among thousands of microbial genomes as well as phylogenetic trees. It also includes a robust multi-aligner for aligning core-genome, parsnp and Gingr which together enables it to generate analysis results, formerly not possible with whole-genome aligners. Along with alignment and visualization, Harvest can also be used for generic file format conversions among standard bioinformatics file formats.

2.1.8 GET_HOMOLOGUES

GET_HOMOLOGUES [16] is an open source software package for analysis of microbial pan-genomes. It is based on orthology-calling approaches which makes it highly customizable and a comprehensive approach to pan-genome analysis of microbes, available for non-bioinformaticians. It uses multiple clustering algorithms including COGtriangles, OrthoMCL and bidirectional best-hit for clustering homologous gene families. HMMER3 package is used to scan protein domain decomposition for adjusting the clustering stringency.

2.1.9 PanCake

PanCake [17] is a data structure for pan-genome and other related genome sets. Unlike many other pan-genome analysis tools, including **PGAT** [11] or EDGAR,

PanCake [17] is not dependent on gene annotations. Rather it combines similar sequences based on their common features, which are derived from pairwise sequence alignments of genome set under consideration. The **PanCake** [17] data structure has comparatively less space complexity even less than the size of input data set. Nevertheless, comparison of identified core and singleton regions shows good agreements.

2.1.10 PanGP

PanGP [18] is a robust tool for computational analysis of bacterial pan-genome profile. For user interaction, a user-friendly graphical user interface has been provided to graphically demonstrate the pan-genome profile analysis. It is comprised of two basic kind of algorithms. The purpose is to optimize the time-cost of analysing the pan-genome profile against set of populations of multiple bacterial strains, of different sizes ranging from population of dozens to hundreds of genome strains in minimum time frame.

2.1.11 PANNOTATOR

PANNOTATOR [19] is a web based pan-genome analysis pipeline, a well-suited software package for automated high quality annotations and pan-genome analysis of closely related genome set. The aim of this pipeline is to reduce the manual work by automating the analysis report generation and correction of various ordering genomic strains. As result of an annotation transfer, it has scored an accuracy of 98% and 76% for gene name and function, respectively. In comparison with a gold standard annotation against the same species, it has achieved similarity level with significance of 70%. These results has surpassed the RAST and BASys software by 41, 21% and 66, 17% for gene naming and function annotation, respectively. **PANNOTATOR** [19] facilitates a reliable and robust pan-genome annotation, allowing the user to focus on the research on genotyping between strains.

2.2 Available gene/protein Databases

All published and sequenced genomes are available on internet. It is a core requirement of every bio-related journal that all published and sequenced DNA, RNA or protein sequences must be added in a publicly available database. The three main and large databases including the NCBI database[1], the European Molecular Biology Laboratory (EMBL) database[2], and the DNA Database of Japan (DDBJ) database[3] are the major resources for accessing genes. These databases collect all publicly available DNA, RNA and protein sequence data and make it available for free. The available gene/protein databases are as follows:

- GenBank [20]
- UniProt [21]
- Swiss-Prot [22]
- TrEMBL [22]
- Pfam [23]
- Ensembl [24]

In this subsection, publicly available gene/protein databases used in literature will be discussed.

2.2.1 GenBank

GenBank [20] is the NIH genetic sequence database. It is a collection of all publicly available annotated gene sequences. It is part of the International Nucleotide Sequence Database Collaboration, which comprises the DNA DataBank of Japan (DDBJ), the European Molecular Biology Laboratory (EMBL), and GenBank at NCBI. All these organizations synchronize their data daily.

[1]http://www.ncbi.nlm.nih.gov/
[2]http://www.ebi.ac.uk/embl/
[3]http://www.ddbj.nig.ac.jp/

2.2.2 UniProt

The **Universal Protein Resource** (**UniProt**) [21] is a comprehensive resource for protein sequence and annotation data. The **UniProt** databases consists of the UniProt Knowledgebase (UniProtKB), the UniProt Archive (UniParc) and the UniProt Reference Clusters (UniRef). It is a collaboration between the the SIB Swiss Institute of Bioinformatics, the Protein Information Resource (PIR) and European Bioinformatics Institute (EMBL-EBI) which are involved through different tasks such as database curation, software development and support.

2.2.3 Swiss-Prot

SWISS-PROT [22] is a protein sequence database. Its purpose is to provide a high level of annotation protein sequences with multiple meta information including function description of proteins, protein domains structure etc. It ensures minimal redundency level and higher integration level with other various gene or protein databases. Latest developments in the database are related to structure including its format and content optimization, improvements in documentation and content references to other databases.

2.2.4 TrEMBL

In 1996, **TrEMBL** (Translation of EMBL nucleotide sequence database) [22] was announced. This database consists of computer-annotated entries derived from the translation of all coding sequences (CDSs) in the EMBL database, except for coding sequences already included in **SWISS-PROT** [22].

2.2.5 Pfam

The **Pfam** [23] is a large protein database. It is a collection of protein families in which each family is represented using hidden Markov models (HMMs) and multiple sequence alignments. Each protein is consists of one or more functional regions, usually known as domains. Different combinations among multiple domains give rise to a new protein having different functional domains. These characteristics also provides valuable insights of protein. This database also generates clans which is a high level clustering of related protein families.

2.2.6 Ensembl

The **Ensembl** [24] database provides a framework consists of the sequences of large genomes as well as a comprehensive source of automatic and stable annotations of genome sequences of human specie. It is a authentic source of gene prediction and integrated with other bio-data sources. It is an open source software development project. The scope of this project is to develop a scalable as well as portable system able to handle large genome data deluge, its sequence analysis and interactive visualization.

2.3 Computational steps involved in pan-genome analysis

Computational pan-genomic analysis involves complex computations. The whole process is sub-divided into small sub-processes assembled in a pipeline to obtain the required results as shown in figure 2.1.

1. Pre-processing
2. Gene/protein comparison
3. Clustering
4. Report generation

The proceeding sections will explain in detail, the computational steps involved in pan-genomic analysis.

2.3.1 Pre-processing

Pre-processing includes steps that involves obtaining input sequenced genomic data from different public databases or user defined query, its validation and conversion from native format into fasta formatted files (if required). The accuracy of

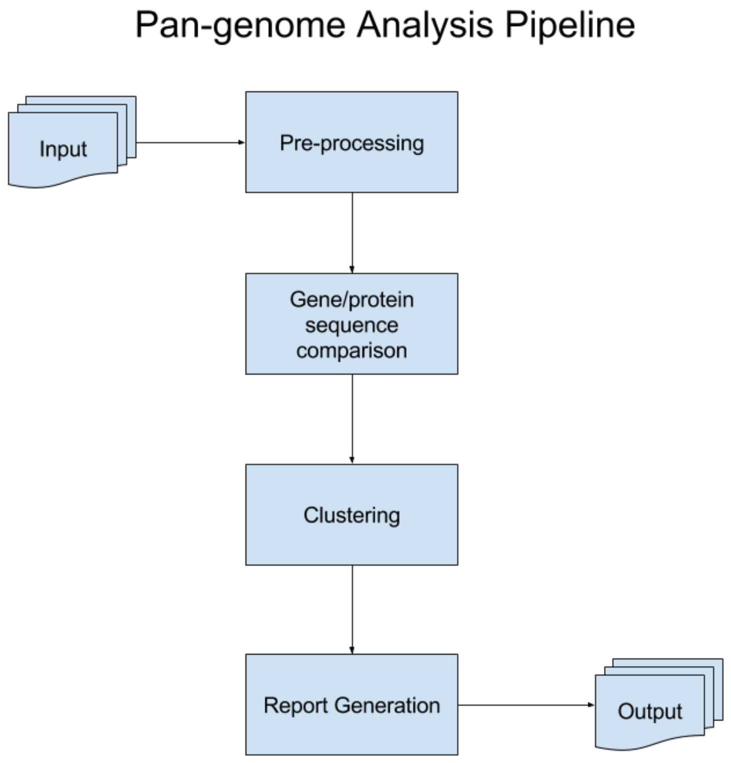
Pan-genome Analysis Pipeline
Input
Pre-processing
Gene/protein sequence comparison
Clustering
Report Generation
Output

Figure 2.1: Step involved in Computational Pan-genome analysis.

finally generated report and efficacy of proceeding steps depends upon the preprocessing step, therefore pre-processing step has a vital role in the overall efficiency of computational pan-genome analysis. Along with data validation of input sequenced genomes, pre-processing step is the stepping stone for the consolidation of already compared sequenced genomes result data, to be used by latter queries for gene/protein comparison and clustering purposes.

2.3.2 Gene/protein comparison

After pre-processing, gene/protein comparison is the next step as well as a prerequisite of clustering. The primary objective of gene/protein comparison is assessment of similarity between the sequenced genomes in the given user-defined query data or with the database as well. It performs sequence alignment between homologous regions. It joins together these aligned sequence regions into larger alignments. This step ensures the gene-gene/protein-protein sequence comparison results with confidence of 50% or more sequence similarity among the compared genomic sequences.

2.3.3 Clustering

Clustering is a step of combining homologous sequences on the basis of self-similarity records (Berry and Linoff, 2004). The complexity involved in individual gene analysis can be largely reduced by grouping genes with similar characteristics into same clusters. It gives a birds-eye-view of the complete genomic data that is essential in domain analysis of proteins and comparative genomics studies [25]. Once appropriate clusters are achieved, usually it is possible to locate distinct patterns within each cluster, making it possible to search information regarding domain similarities and gene interaction. Several types of clustering algorithms and methods have been developed to improve the clustering efficiency of genomic data (Eisen et al., 1998; Tamayo et al., 1999; Dhaeseleer et al., 1999; Friedman et al., 2000; Holmes and Bruno, 2000; Jact et al., 2001; He et al., 2003; Kasturi et al., 2005). Taking the result of gene/protein comparison step and grouping the finally sequenced genomes/proteins into clusters based on their relative characteristics. Clustering is an important step as it rapidly decrease computation involved in the final pan-genome and core/unique gene extraction step.

2.3.4 Report generation

Finally, gene/protein clusters are used to extract the core genomes/proteins, unique/strain-specific genomes/proteins, dispensable genes/proteins and pan-genomes. The final report is comprised of multiple formats including numeric tables and multiple visual graphs to ease the understanding and comparison of computational results including number aggregate as well as the computationally resulting sequences against laboratory output.

2.4 Reference Implementation

The CMG-biotools [26] package is our reference implementation to PanCGP pipeline. The tool has been designed for bioinformaticians in a very basic knowledge of UNIX and limited introduction to computational analysis. It enables the user to perform multiple computational analysis with a list of genomes or proteins. The operations include structure analysis of DNA, proteome sequence comparison, phylogenetic analysis. The package also supports FASTA format sequences from unpublished sources. This software package facilitates the user with a stand-alone command line interface for microbial analysis. The first step is to get the genome data set for an organism or set of organisms. The next step is to find the coded regions in DNA sequences. Some genome datasets are curated manually along with high-quality annotations but others may not be annotated at all. So, for annotating others, CMG-biotools [26] uses a gene finding algorithm without any further evaluation of resultant information. The package uses the program Prodigal [27] for the purpose of gene finding which is included in the gene finding pipeline known as prodigalrunner. This pipeline takes FASTA/multi-FASTA file as input or genome DNA sequences. The output consists of four files:

1. A GenBank file with extension as *.gbk*.
2. A FASTA format frame file with extension as *.orf.fna*.
3. A FASTA format protein sequence file encloses gene translation with extension *.orf.fsa*.
4. And a general feature formatted file with extension as *.gff*.

A BLAST matrix is used to visually represent the pairwise comparison of proteome using BLAST (Basic Local Alignment Tool) [6, 7]. All given sequences are compared reciprocally to each other. A BLAST hit of 50% is considered significant, i.e. at least 50% of alignment results in an identical match. Sequences having similarity with 50% cutoff are considered as one protein family. For building protein families, single linkage clustering is used. For pan-genome analysis, BLAST results are used to categorize the overall characteristics of input gene dataset. The core-genes are the common set of gene found in all the genomes under consideration. The pan-genome is the complete set of protein families identified in the investigated gene set. The shared genes are common to more than one gene families but are uncommon to given genomes in the set. During analysis, the first genome is equal to both core-genome and pan-genome. With the addition of more genomes, the size of core-gene gradually decreases or remain constant based on the percentage of match found with the incoming genome. If a new gene sequence matches, it becomes a part of core-genome family, if it doesnt, if becomes a new protein family. The size of pan-genome keeps on increasing with the addition of new genes. For pan-genome analysis, pancoreplot pipeline is used which is included in the CMG-biotools [26] software package. It generates a plot of the underlying pan-genome analysis results calculated against given gene/protein sequence data set.

2.5 High Performance Computing

High Performance Computing is generally known as the practice of combining the computational power of existing computing resources in such a way that delivers much higher performance than a single desktop computer or workstation in order to solve complex, computationally expensive problems in science, engineering and business. The term applied to the systems capable of operating at 1012 or more floating point operations per second.

2.5.1 Amdahl's law

One reason for less than perfect speedup is due to the code or part of code can be inherently sequential which restricts the efficiency of code. Let's take an example that 10% of the whole code is sequential. This enlightens the fact that no matter the number of available computing resources, the sequential part of code will take its time to complete and limit of execution speedup of whole code by a factor of 10.This phenomenon is called as Amdahl's Law [28]. This is formulated as follows.

$$T_p = T_1(F_s + F_p/P) \tag{2.1}$$

Where F_s denotes the sequential fraction and F_p denotes the parallel fraction (or more strictly: the *parallelizable* fraction) of the code, respectively. T_p represents the total parallel execution time on p number of processors.

In case of hybrid programming, i.e, the mixture of shared-memory and distributed memory programming, the Amdahl's law states that p nodes with c cores each, and F describes the fraction of the code that uses c-way thread parallelism. It is assumed that the whole code is fully parallel over the p nodes. The ideal speed up would be pc, and the ideal parallel running time T1/(p*c), but the actual speed-up time will be:

$$T_{p,c} = T_1(F_s/p + F_p/pc) = T_1/pc(F_sc + F_p) = T_1/pc(1 + F_s(c - 1)) \tag{2.2}$$

In Amdahl's law, $1/F_s$ limit the sequential portion speedup whereas in hybrid programming it is restricted to p/F_s by the task parallel portion.

2.5.2 Scalability

In case of parallel code execution, either the number of computing cores are matched with the problem size or an increasingly complex problems are ported to growing number of computational resources. Both cases hardly satisfy the concept of speedup. Rather, the concept of scalability is used.

Scalability is of two types. One is strong scalability and the other is weak scalability. Strong scalability is same as speedup as well as discussed above, that a code is strongly scalable if it shows perfect or nearly perfect speedup after scaling to more and more computational resources.Normally, claiming scalability upto 'n' number of processors shows that the code will be noticeably shows perfect speedup upto 'n' number of processors. In terms of efficiency, it is not necessary for a problem to fit on a single processors: often a smaller number such as 64 processors is used as the baseline from which scalability is judged.

More interestingly, weak scalability is an ambiguous term. It states that, if the problem size and number of computing resources increases such that the number of operations per resource remains the same, the speedup in operations per second against each processor also remains the same. This measure is rather hard to report. The results may vary depending upon the relation between the number of operations per processor and the amount of data. Suppose that the relation is linear, the amount of data per computing resource/processor will be same and the parallel execution time will be constant as the number of processors increases. .

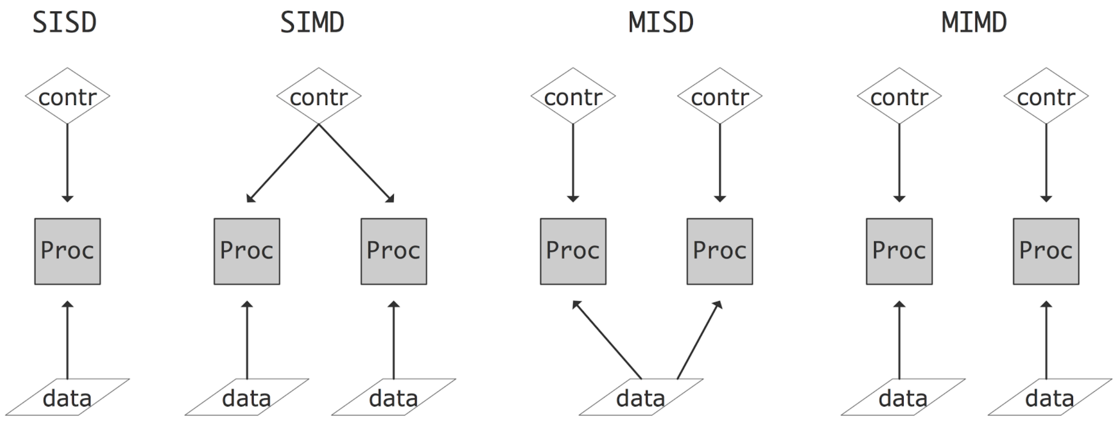

Figure 2.2: The four classes of Flynn's taxonomy [29]. **(1) SISD, (2) SIMD, (3) MISD, (4) MIMD**

2.5.3 Parallel Computer Architectures

Super Computers fall in a category of parallel computers, i.e, an intelligent hardware that allows the parallel execution of multiple instructions or instruction sequences. Various forms of simultaneous execution of multiple instructions to take place is characterized by Flynn [29]. Flynns taxonomy characterizes architectures by whether the data-flow and control-flow are shared or independent.

Single Instruction Single Data (*SISD*): this is the conventional CPU architecture: only one instruction at a time can be executed which can operate only on a single data item.

Single Instruction Multiple Data (*SIMD*): in this type of computer architecture, there can be multiple processors and each processor is operating on its own data. In *SIMD*, all processors are executing the same instruction on same data.

Multiple Instruction Single Data (*MISD*): No existing architecture follow this rule; it states that in multi-processor environment, each processor is executing instruction on a single data item. Redundent computations for safety-critical applications are an example of MISD.

Multiple Instruction Multiple Data (*MIMD*): According to this architecture, multiple processors execute instructions on multiple data items, every execution is independent of other. Most current parallel computers follow this architecture.

2.6 HPC in Bioinformatics

Latest research in the field of biochemistry and biotechnology as well as the breakthroughs in high performance computing and computational modelling has made developments in healthcare sector, drug discovery, genome identification and discovery and systems biology. Combining these development breakthroughs together has made a healthier life towards better living possible by inventing new therapeutic strategies. As a result of these effort, new fields have been emerged such as Computational Biology and Bioinformatics. Bioinformatics stretch to a number of research fields including life-saving biological discoveries has been made possible using complex scientific application, benefiting significantly from increasing computational resources. Normally, hpc techniques are being used for computing biological data, impossible to be computed intime by using traditional computing techniques.

2.7 Java for High Performance Computing

After its first release in 1996, Java became highly popular for developing business and scientific applications. According to various computer scientists, Java could be a good alternative for developing scalable and robust high performance computing application. As compared to C programming language and FORTRAN, java programming language became popular as it comes with multiple advantages including high-level object oriented programming concepts, automated memory management and heap garbage collection, efficient compilation and runtime checking, improved debugging and platform independence which means it has support for porting it to any hardware or operating system, provided that an installation of Java Virtual Machine (JVM) exists on that system to host java-based applications. This significant contribution of Java Virtual Machine enabled programmers to focus on the application related domain conversion issues and domain of interest rather than issues related to the heterogeneous nature of underlying architecture.

2.8 MPJ Express

MPJ Express [30] is a high performance computing library based on java that enables high performance computing in an application. Application is programmed in java using JavaMPI. It supports nested parallelism by using Java OpemMP by parallelising computation within an MPJ Express process. It seamlessly divide the computation on distributed as well as shared memory architectures. The main focus of MPJ Express [30] is to minimize the communication overhead in the parallel software applications and provide a flexible mechanism to enable high performance computing in applications with performance cost as minimum as of its C counterparts.

Chapter 3

Proposed Methodology and Implementation

3.1 Pan-genomic and Comparative Genome analysis Pipeline

Next generation sequencing techniques are playing a vital role in pan-genome data extraction and identification of different isolates according to their relative properties. The proposed pipeline is divided into six steps: input FASTA decomposition, sequence hashing, database creation, sequence comparison, clustering and report generation. This chapter discuss the detailed methodology proposed for Pan-genomic and Comparative Genome Analysis pipeline tool. Figure 3.1 demonstrate the overall pipeline for pan-genomic analysis.

3.2 Test data sample

We utilize 38 strains of *H. Pylori* shown in Table 3.1. *H. pylori* microaerophilic organisms significantly lives in the stomach. An Australian researchers Barry Marshall and Robin Warren found in 1982 that it was available in an individual's with incessant gastritis and gastric ulcers, which were not already connected with microbial cause. It is likewise connected to the advancement of duodenal ulcers and stomach tumor. It has an extensive difference regarding strains with a completely

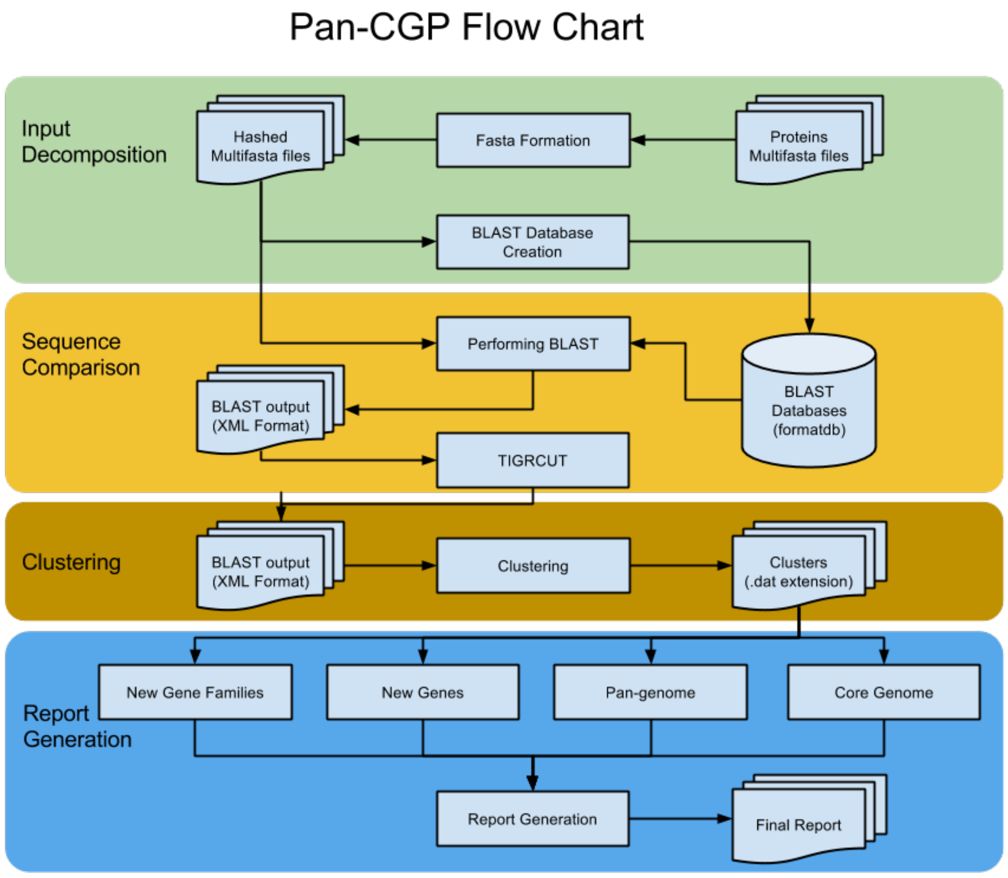

Figure 3.1: Flow chart diagram of PanCGP pipeline

sequenced set of genomes [8, 31]. The strain ”26695” genome comprises of about 1.7 million base sets, making upto total of 1,576 genes. The two strain sequences indicate substantial genetic contrast, with up to 6% of the genetic diversity. H. pylori has 5 largeouter membrane protein families [32] among which putative adhesins is the largest of all. The remaining four includes flagellum-associated proteins, iron transporters and other proteins with obscure domains. A characteristic of *H. Pylori* similar to other commonly found gram-genative micro-organisms, its outer membrane is composed of phospholipids and lipopolysaccharide (LPS) as well as cholesterol glucosides that may found in couple of other bacteria.

Table 3.1: Sample dataset consisting of 38 bacterial strains of Helicobacter pylori

Serial No.	**Strains**
1	Helicobacter_pylori_2017
2	Helicobacter_pylori_2018
3	Helicobacter_pylori_26695
4	Helicobacter_pylori_35A
5	Helicobacter_pylori_51
6	Helicobacter_pylori_52
7	Helicobacter_pylori_83
8	Helicobacter_pylori_908
9	Helicobacter_pylori_B38
10	Helicobacter_pylori_B8p
11	Helicobacter_pylori_Cuz20
12	Helicobacter_pylori_ELS37p
13	Helicobacter_pylori_F16
14	Helicobacter_pylori_F30p
15	Helicobacter_pylori_F32p
16	Helicobacter_pylori_F57
17	Helicobacter_pylori_G27p
18	Helicobacter_pylori_Gambia94_24p
19	Helicobacter_pylori_HPAG1p
20	Helicobacter_pylori_HUP-B14p
21	Helicobacter_pylori_India7
22	Helicobacter_pylori_J99
23	Helicobacter_pylori_Lithuania75p
24	Helicobacter_pylori_P12p
25	Helicobacter_pylori_PeCan18
26	Helicobacter_pylori_PeCan4p
27	Helicobacter_pylori_Puno120p
28	Helicobacter_pylori_Puno135
29	Helicobacter_pylori_SAfrica7p
30	Helicobacter_pylori_Sat464p
31	Helicobacter_pylori_Shi112
32	Helicobacter_pylori_Shi169
33	Helicobacter_pylori_Shi417
34	Helicobacter_pylori_Shi470
35	Helicobacter_pylori_SJM180
36	Helicobacter_pylori_SNT49p
37	Helicobacter_pylori_v225dp
38	Helicobacter_pylori_XZ274p

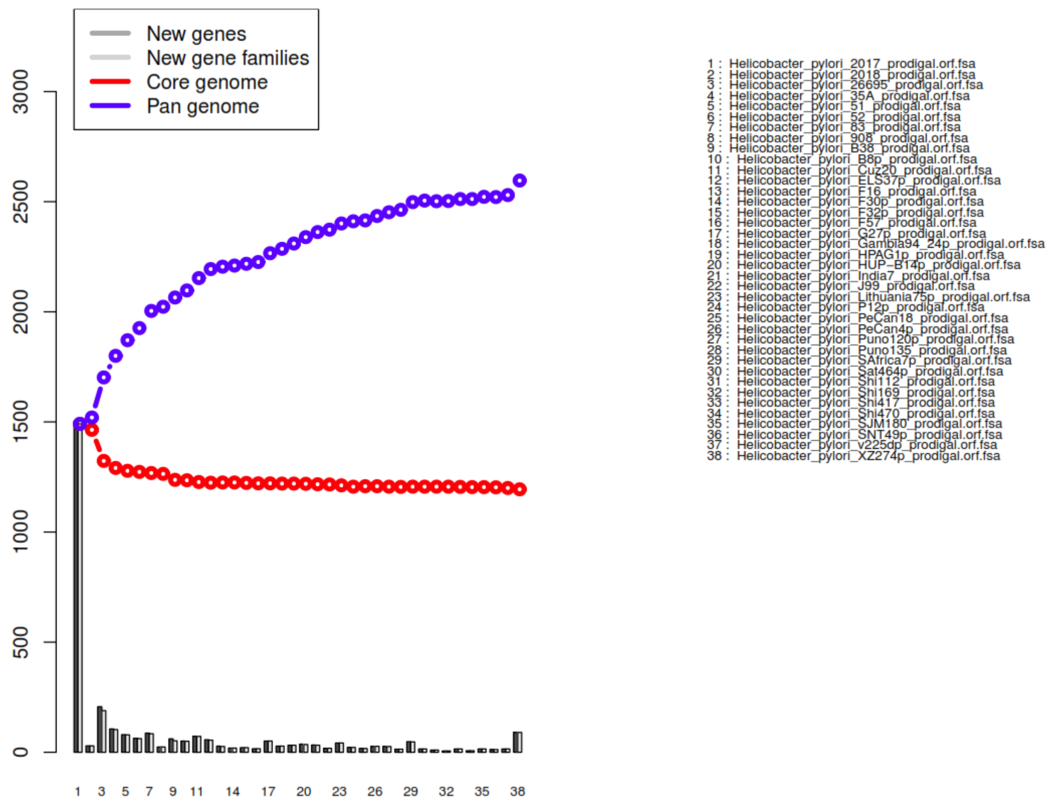

Figure 3.2: Core and pan-genome graph using pancoreplot pipeline in CMG-biotools [26]. *Pan-genome* keeps on increasing with increasing number of genomic strains. *Core genes* keeps on decreasing with increasing number of genomic strains as common genes keeps on decreasing.

We utilize GenBank database for the sample gathering purpose and using the same data for benchmarking the both old and newly proposed pan-genomic analysis pipeline. It is in the form of multi-fasta files containing the protein sequences along with their unique identifier to provide as an input to the suggested pipeline tool.

3.2.1 Benchmarks

We take 38 strains of Helicobacter Pylori as our benchmarking datasets shown in the Figure 3.2. And run CMG-biotools [26] pan-genome analysis to extract core genome/proteins, unique genomes/proteins and shared proteins among different protein sequences. Parsing the FASTA files, hashing the input data to be able to process the unregistered dataset as well, protein-protein BLASTing for more than 50% confidence match among the sequenced proteins, clustering the data and parsing the results in the form of final report graphical as well as non-graphical.

3.3 PanCGP tool design

PanCGP is being developed as parallel high performance computing as well as scalable pan-genomic analysis tool. It is designed to speed-up the processing time took by other conventional pan-genomic analysis tools in processing the pan-genomes on a set of given data. It intelligently divide the problem into smaller chunks and run the operations in parallel on all those chunks to get results in optimal time. It also applies task parallelism by running those tasks parallel that are independent of one another so that maximum output is achieved in a comparatively less time. It is adaptive to the number of available resources (computing nodes), i.e. it can scale depending on the availability of resources ranging from a personal computer to high performance supercomputing clusters running multiple node at a time to accomplish complex computing tasks. The architecture of pipeline is highly modular with a very low degree of inter-modular coupling which ease the process of alteration in pipeline. The architecture diagram of *pancgp* is shown in figure 3.3. In case of updating the pipeline from future perspective or integration of extra modules is highly convenient. This results in a very low integration effort in case of update or any changes afterwards that needs to integrate modules in the *pancgp* pipeline.

The whole pipeline is comprised of following modules:

1. Input FASTA decomposition
2. Sequence hashing
3. Database creation
4. Sequence comparison
5. Clustering
6. Report generation

3.3.1 Input FASTA decomposition

The 38 strains of Helicobacter pylori is given as input to the pipeline. To increase the efficiency and reduce the cost of input, high performance computing played

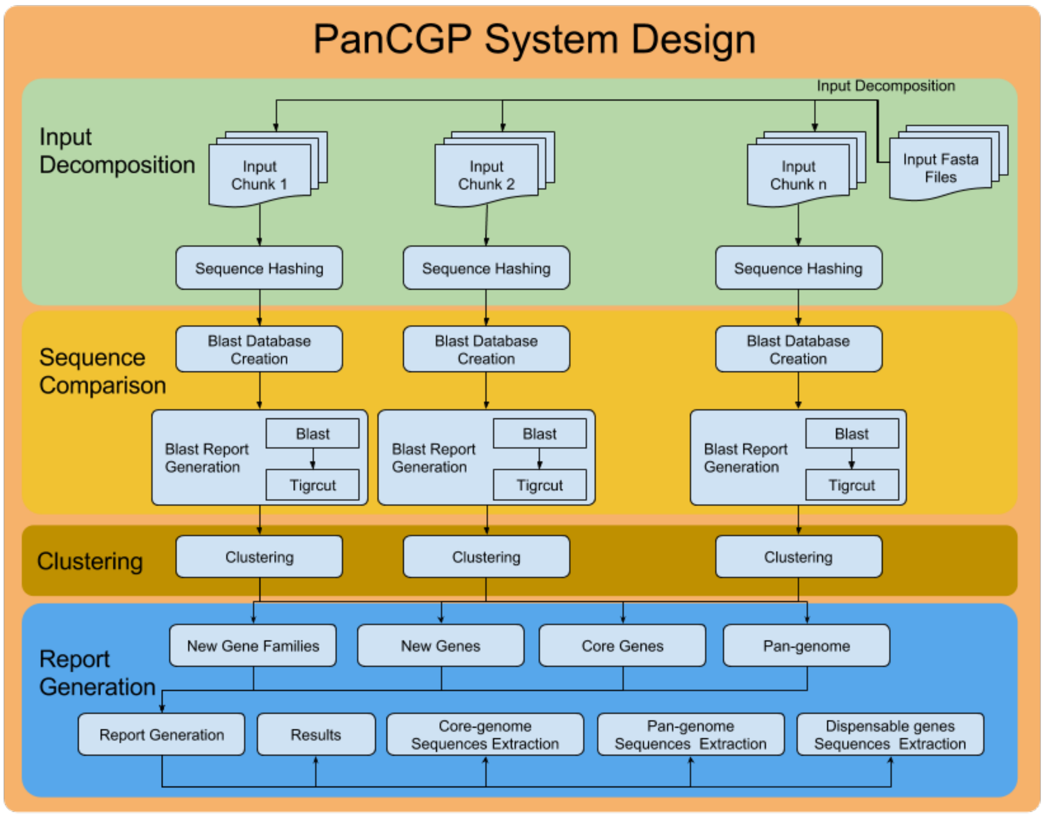

Figure 3.3: PanCGP parallel system design model

Data parallelism
Task parallelism
Range of parallel algorithm

Figure 3.4: Categories of parallelism: (1) Data parallelism. (2) Task parallelism. All parallel algorithms falls within these two extremes

a vital role. As described in Figure 3.4, Parallelism falls in two categories: data parallelism and task parallelism. Every parallel algorithm lies in between these two extremes. To reduce the time cost of our pipeline, by using the data parallelism technique results in increasing throughput of the system and lessen the I/O time. By dividing the large input into small chunks and reading those chunks in a parallel fashion at same time greatly reduce the time cost and increase the overall efficiency as well as efficacy of the proposed pan-genomic pipeline tool.

The given dataset which is composed of multiple strains of *H. pylori*. All the 38 strains of Helicobacter pylori are fed as input to the pipeline simultaneously. The input decomposer decides the size and number of chunks that should be made of given input that suits the optimal task completion. For 38 strains, maximum of

38 parallel inputs can be made, i.e, maximum number of chunks are equal to the total number of input strains to be given as input to the pipeline and smallest chunk size is equal to one strain that can be given to the pipeline for analysis.

While running the pipeline the maximum scalability factor or maximum number of resources are mentioned on which the pipeline is desired to run. The complete syntax for running the pipeline is:

pancgp -tp <value> -datapath <path>

Program Name: *pancgp*

Table 3.2: PanCGP argument list

Agruments	**Description**
tp	Total number of threads to run.
datapath	Path to the folder/container containing the input data source.

pancgp is the name of program or command to run the program. An executable script that takes care of number of resources to engage as well as the path to the source input containing biological sequences to be analyzed. The result is generated in a subfolder of the source input container by giving it a specific name generated by the program from the input data source. The program takes multiple arguments as shown in table 3.2. *tp* is total processors to be taken into account on which program is allowed to execute in parallel fashion. Scaling the jobs on multicore and multinode architecture is seamlessly handled by the *pancgp* itself. For example, user enter a value 8, which describes that maximum 8 jobs can be run in parallel. If available resources are more than 8, the job will be divided to 8 processing resources as follows:

Total available nodes = 6;
Total processing cores per node = 2;
Scalability factor entered by user = 8;
Resources occupied by *pancgp* = (Scalability factor entered by user / Total processing cores per node)
= 8/2 = 4.

So, the total nodes occupied by the *pancgp* program will be 4 nodes. The input data will be divided according to the scalability factor given by the user and the other two nodes will remain idle.

If user is unable to provide the scalability factor, the program itself decide the scalability factor depending on the available resources or computing power as well as the amount or size of input data source. Taking the example given above, if total number of biological strains are equal to or greater than 12, the system will automatically scale on all the available nodes and divide the input into multiple chunks accordingly. In this case, the *resources occupied by pancgp* has the value equal to or greater than *the product of total available nodes and total processing cores per node.*

3.3.2 Sequence Hashing

Once the input is divided into chunks of smaller sizes, it is read and parsed. Sequence hashing is a step where each sequence of every strain given in input data source is hashed for its unique identification throughout the pipeline. The input supported by *pancgp* is FASTA or multi-fasta protein sequence file format. Fasta format has information about each sequence that includes:

1. Header, starts with '>' sign and contain information including the sequence name and other information that helpful in uniquely identifying that particular sequence.
2. Sequence itself that comprises of a consecutive sequence of alphabets each representing a single amino acid.

After dividing input into chunks, input is filtered for empty FASTA entries. Sometimes, FASTA or multi-FASTA formatted files have entries which includes header line but there is no corresponding sequence to that header. Such entries are removed before hashing the sequence as empty FASTA entries can take place in hashing leaving very low complex regions and generating anomalies in sequence comparison results (which is next step after sequence hashing in which sequences are compared and matched based on their similarities and dissimilarities). After removing these zero-length FASTA entries, each sequence related to a strain is stored in a file named with a unique serial number assigned to that strain for its unique identification along with its total number of sequences and a common header but common to all of its sequences. It helps in conveniently iterating through different strains and their related sequences as given in input as well as differentiating sequences of a specific strain from the sequences of other strains.

Sequences of each strains are then hashed. The resulting hashed string is assigned to the respective sequence as header which uniquely represent its respective sequence throughout the process until the final report generation. All the sequences are sorted based on their unique hashed string header.

Since the process is applied to each strain file and completely independent from any other strain file being hashed. So, all strains sequence are hashed using high performance computing approach in a parallel manner. The number of strain files being processed at a time is dependent on the number of available resources and total number of active parallel computing threads.

3.3.3 Database Creation

Once FASTA sequences are hashed, BLAST databases are created. Multiple resources including National Centre for Biotechnology Information (NCBI) as well as Computational Biology Research Group (CBRG) etc, provide databases related to many sequenced genome/protein group to be used for comparison purposes with given sequences as test dataset.

Along with already available BLAST databases, a custom BLAST database can also be created based on our custom genome/protein data against which comparison can be made. For that purpose, index BLAST database files are generated using all strain files containing related hashed sequences named with a serial number assigned to that strain for its unique identification throughout the pipeline and named with the same unique serial number.

The program used to construct the index for BLAST database is $formatdb$. It takes multiple arguments for database creation as shown in table 3.3. The syntax is as follows:

formatdb -i <strain filename> -p <sequence type> -t <target filename>

Program Name: $formatdb$

PanCGP intelligently scales and optimizes the whole process depending on the available computing resources and the number of input strain files to increase the efficiency and efficacy of the whole process. As mentioned above, a dataset

Table 3.3: formatdb argument list

Agruments	**Description**
i	Input filename containing genome/protein sequences.
p	Type of sequence. 'T' indicates protein sequence.
t	Target filename containing BLAST index against given input file sequences.

of 38 strains of Helicobacter pylori taken as test data and the total number of available computing resources are twelve (total available computing nodes times the computing cores per available computing node) Then:

Number of iterations = Ceil(total number of inputs / total number of computing threads)
= 38/12 = 4 iterations.

So, each thread will make four iterations, treating four strain inputs respectively, to compute the indices for BLAST database that will be used in sequence matching.

Finally, the BLAST database index created is splitted in chunks stored in multiple files, each related to a specific strain. This facilitates less time consumption with efficient use of available resources. At the completion of database formation step, three files naming "$targetname.fsa.pin''$", "$targetname.fsa.phq''$" and "$targetname.fsa.phr''$" are created against each strain file containing the BLAST indices used by BLAST [6, 7] in the next step for sequence matches.

3.3.4 Sequence Comparison

After creating BLAST databases, next step is sequence comparison in which all sequences are compared based on their similarity ratio. A BLASTP comparison tool from NCBI is used to compare the sequences of proteins. A reciprocal BLAST is a computational technique used to identify the homology within the genome/protein sequence strains. In this method, two sequences are blast with each other such that first sequence, is used as a query sequence with the BLAST database of the second sequence. Then second sequence is blasted as a query sequence with the BLAST database of first sequence. In both cases, results may differ depending on the similarity of first genome/protein sequence toward the second genome/protein sequence and vice versa. The complete syntax of BLAST [6, 7] is:

blastall -F <filter query sequence type> -i <input sequence> -p <blast type> -e <expectation value> -m <blast output format> -d <database filename> | TIGR-CUT | gzip > <output filename>

Program Name: *blastall*

Table 3.4: BLAST [6, 7] argument list

Agruments	**Description**
F	Filter query sequence. Can be either true or false.
i	Input query sequence.
p	Type of blast program to run.
	(In our case *blastp* is used for protein sequences.)
m	Type of blast output format.
t	Blast database index filename.

The program takes multiple argements as shown in Table 3.4. The output of blast [6, 7] is in xml format which is fed to TIGRCUT as input. TIGRCUT ensures the 50-50 rule in which there is a 50% cutoff identity and 50% ALR on blast output, i.e, only those results are filtered out in which comparison results meet the similarity percentage of at least 50%. Less than that are discarded. The final results are compressed and written to a separate file.

PanCGP runs multiple blast programs at a time depending on the number of computing threads as well as the availability of resources. Since, the database created is distributed blast database where every strain of the specie is separately indexed and leaving our problem in hand as embarrassingly parallel problem which is an ideal scenario in parallel programming paradigm. So, while blasting this distribution results in independent multiple blast process to run at the same time allowing the pipeline to be high parallel and scalable which results in the high throughput and enhanced usability from a single node to cluster architecture.

3.3.5 Clustering

After BLAST process exits and sequence data is filtered out based on similarity percentage, clustering is applied to group the similar sequences into families. Clustering is a process of grouping similar objects in the form of a cluster. Protein clustering refers to the clustering of homologous proteins into clusters. Proteins grouped in a cluster or family have comparatively more similar traits as compared to other protein sequences.

The clustering process is done by taking a single sequence and start comparing it with every other sequence of all strains. A transitive closure of given protein sequences is formed where the single sequence taken as query act as a representative sequence of the cluster being made. A threshold is set and those sequences having similarity level within threshold are grouped in a cluster or family. The similarity is measured based on the sequence alignment results from BLAST [6, 7] or sequence comparison step. If a sequence falls of the threshold then that sequence become the representative of a new family or cluster. Same process is repeated with the rest of sequences until the last sequence is reached.

As discussed above, the comparison result between two may be different when compared with one another and vice versa. This makes the clustering step more compute intensive since all reciprocal results have to be analyzed to group any two sequences in a cluster. It takes multiple arguments to get the path of blast result files and unique serial number related to every strain to identify the strain as listed in table 3.5. The syntax is:

Grouping -dirpath <path to blast result files> -rank <unique serial number related to every strain> <filename of temp file maintaining info. against each strain>

Program Name: *Grouping*

Table 3.5: Grouping argument list

Agruments	**Description**
dirpath	Path to directory containing blast result stored in compressed files.
rank	Unique serial number assigned to each strain for identification.
	(A temporary file maintaining information regarding each strain of query data).

To optimize the performance of clustering in *pancgp*, it reads data from a distributed source as input and also write results following same distributed pattern. PanCGP scales the process and apply data parallelism to process the huge amount of data in an efficient manner within a short time span. It divides the data into smaller chunks and feed as input to multiple clustering processes running in parallel. The degree of parallelism is decided depending upon the number of available resources as well as the size of the given data set. All this decision making is done intelligently by *pancgp* itself. The results generated are also stored in distributed manner. The resulting data is in the form of object files where every file contains the clustering results against a specific strain and named with the unique serial number of that particular strain.

3.3.6 Report Generation

The output of clustering is in the form of distributed object files which needs to be parsed to get the results and present in the form of valuable information. This step is done when all threads that are executing in parallel, each on a small chunk of data, submit their results. Those results are interpreted in the form of *pan-genome, core genome/proteins, dispensable genes/proteins, unique/strain-specific genes/proteins, new gene/protein families and total number of genes per strain* of the specie. This information is extracted and stored in the form of a text file in a tab-separated tabular format in which each strain of a specie along with its all necessary information as well as the information derived using *pancgp* is stored. It also extracts the corresponding protein sequences against *pan-genome, core protein and dispensable protein sequences* in separate text files. The tabular text format can be used to render the final results in the form of visual graphs representing the trends in gene/protein sequence analysis.

Chapter 4

Results and Discussion

This chapter describes the results of PanCGP pipeline and its comparison with the other tools. The evaluation of PanCGP pipeline tool is based on multiple performance and usability metrics including time cost, complexity, scalability and GUI.

Time cost reflects the time taken by the algorithm implemented in the pipeline tool to process a specific task. Complexity of algorithm defines how fast or slow an algorithm can perform, denoted as a numerical function T(n), i.e, time T taken by the algorithm to process an input of particular size 'n'. T is independent of the implementation details. An algorithm may take different amount of time depending on multiple factors including: I/O speed, processing power, compiler version, instruction set and network latency etc. Algorithmic complexity is measured in terms of asymptotes in which we assign a constant time to each elementary step being performed by the algorithm in any particular manner. Scalability factor of a parallel algorithm on parallel or distributed architecture is its ability to scale to a number of available resources effectively. Scalability analysis is used to determine the best algorithm-architecture combination of the problem against a given input.

4.1 Comparison Results

This section discusses the benchmark results of PanCGP using high performance parallel computing technique. The PanCGP pipeline has been benchmarked using a high performance computing facility available at Research Centre for Modelling and Simulation (RCMS) National University of Sciences and Technology (NUST) Pakistan. PanCGP is benchmarked against the sample dataset of 38 bacterial strains of *H. pylori*. The evaluation process is based on performance parameters including speedup, scalability and efficiency. Figure 4.1 shows the benchmark results generated on different number of processors ranging from 1 to n where n is the total number of available bacterial strains for pan-genome analysis.

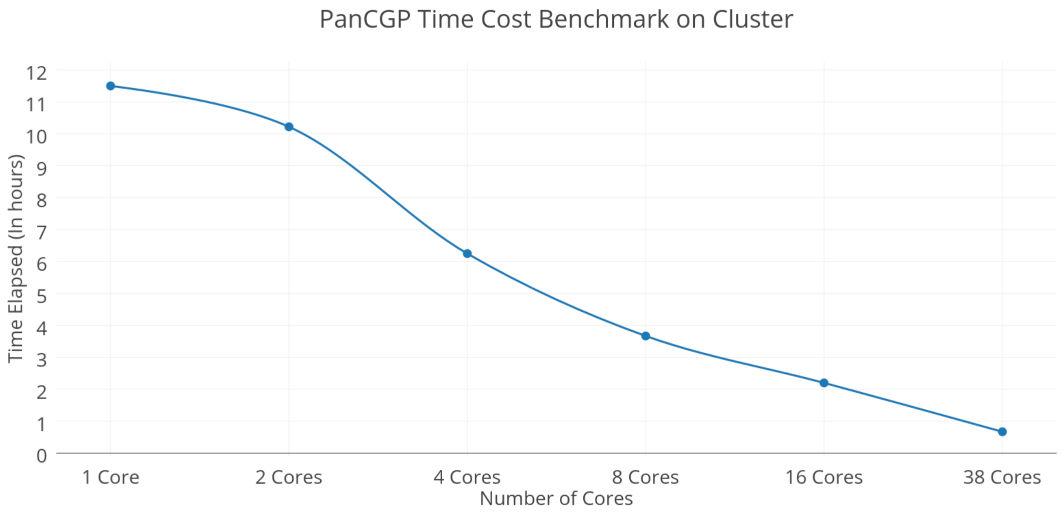

Figure 4.1: Benchmark results of PanCGP pipeline against sample dataset of 38 helicobacter pylori bacterial strains.

4.2 Benchmarks

The graph shown in Figure 4.1 demonstrate the speed as well as the high scalability level of PanCGP pipeline tool. As the number of cores increases, the data is divided in a suitable manner to gain maximum benefit from the underlying processing hardware. On the other hand, CMG-biotools [26] is a stand-alone implementation with a minimum scalability factor. It supports only shared memory architecture to divide its contents across multiple computing devices. Figure 4.2 shows the benchmark results from pancoreplot, a CMG-biotools [26] pan-genome analysis pipeline where the speed up is negligible as compared to PanCGP. The benchmarks has been calculated on a local computer equipped with 3.2 GHz core i7, having four processing cores and 8 logical cores after incorporating hyper-threading.

Drawing a comparison between the two pipeline results shows a considerable speedup and higher scalability of PanCGP pipeline as compared to CMG-biotools

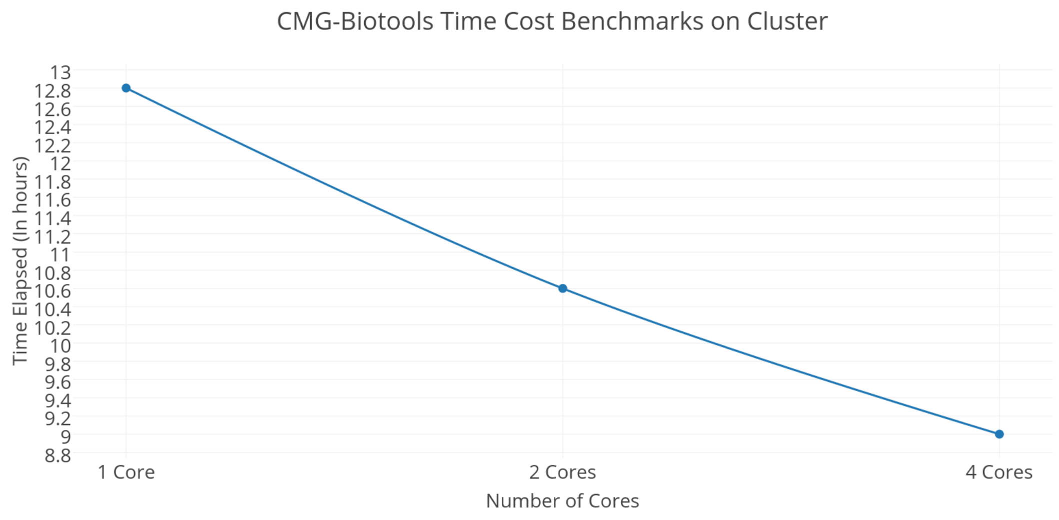

Figure 4.2: Benchmark results of Pancoreplot, a CMG-biotools [26] pan-genome analysis pipeline against sample dataset of 38 helicobacter pylori bacterial strains.

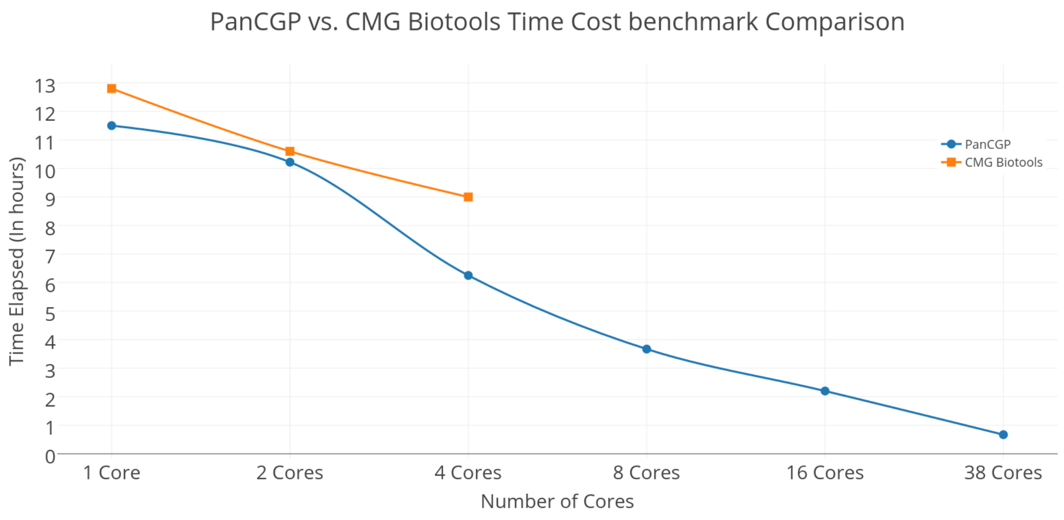

Figure 4.3: PanCGP vs. CMG-biotools [26] pancoreplot pipeline time-cost benchmark comparison

[26] pan-genome analysis pipeline. Figure 4.3 shows the comparison graph of both the pipelines.

4.3 PanCGP vs. CMG-biotools

CMG-biotools [26] is our reference benchmark tool for pan-genome analysis. It is designed for bioinformaticians with a very little or no knowledge about UNIX. It comes as a standalone package to run on a single computing machine. It takes a list of sequences of bacterial strains published as well as unpublished and run pan-genome analysis to generate statistical results in a graphical as well as tabular format. A separate utility is used to extract the sequences act as core-genome and pan-genome. The limitation of CMG-biotools [26] is its level of ease of use and its ability to scale according to the size of input to the number of available resources for results computation in an optimal time frame which makes it inconvenient to use during research projects as it takes a large amount of time to compute results against a particular size input. Due to its scalability limitation, it does not support modern high performance parallel computing clusters based on hybrid architectures, i.e. both shared memory computing architecture as well as distributed memory computing architectures.

PanCGP is a high performance parallel pan-genome analysis pipeline. It is a java based software package which incorporate multiple tools to process protein sequences for pan-genome analysis. It supports both published as well as unpublished sequences of bacterial strains and generate the final pan-genome analysis statistical report in simple tabular format that can be easily parsed to generate multiple graphical outputs. It also generates files containing sequences that makes up pan-gnome and core-genome in separate folders respectively.

Unlike previous pan-genome pipeline, PanCGP is based on modern high performance parallel computing architectures and supports both shared memory architecture and distributed memory architectures as well as hybrid architectures incorporating both shared memory and distributed memory architectures. It seamlessly configure itself according to environment on which it need to execute. In case of local computing machine, it automatically scales on the available processor and divide the tasks intelligently according to the underlying architecture as well as on clusters comprised on multi-node and multi-core systems, it efficiently make use of nested parallelism to scale according to the size of the input data for efficient parallel processing. Robustness and support of modern computing architecture makes it highly convenient to use in research project where timely result generation is the key to innovation.

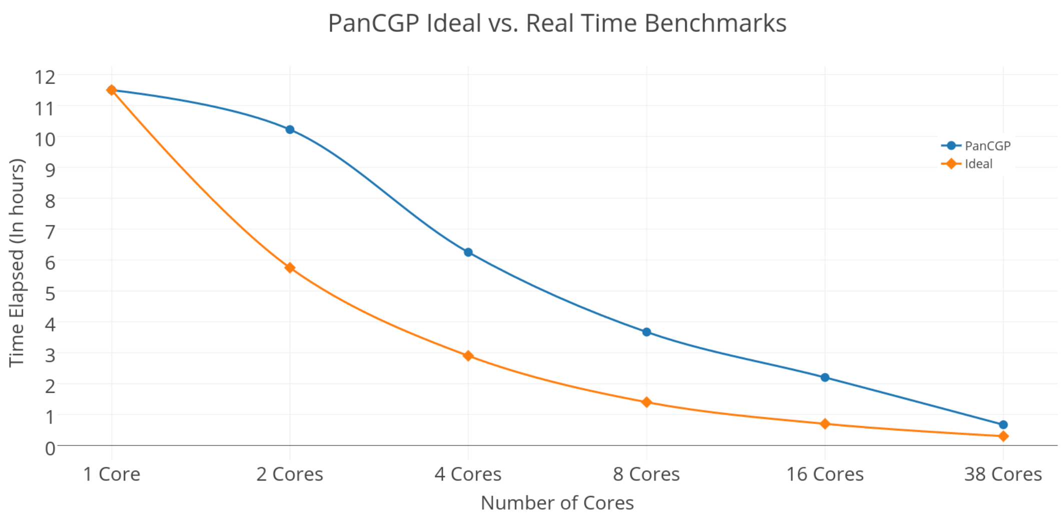

Figure 4.4: PanCGP real benchmarks vs. ideal benchmarks.

4.4 Ideal vs. real time benchmarks

Ideal time is theoretical measure of time taken by the application to complete a specific task whereas the real time is the experimental measure of time taken by the application to perform that specific task. Ideal time of pipeline shows a straight line declined as the number of processors increases, ideally cutting down the processing time by the number of processors being used in parallel for pipeline execution.

Real time took by the PanCGP application during execution may express a deviation from the ideal behaviour while changing the number of processors being used in parallel depending upon the factors affecting the speed of execution of PanCGP pipeline. Figure 4.4 represents the comparison between the trends followed by theoretically ideal execution of pipeline under varying number of processors being used in parallel for execution of pipeline and the experimental execution time in which PanCGP complete its execution under different number of processors being used in parallel for execution of pipeline.

4.5 Factors affecting the PanCGP speed

Experimental results of PanCGP pipeline differ from ideal results due to the factors involved that increase the overhead depending on the applied parallelism for purpose of execution. Factors involved in the deviation of experimental results from ideal results are:

- I/O speed
- Computation overhead
- Process communication overhead

4.5.1 I/O speed

I/O speed can be a major overhead affecting the speed of PanCGP pipeline. It depends on the device on which I/O operation are being taken place. Memory I/O speed is much higher than an external or secondary storage disk. The gap between memory speed and average disk I/O speed of access stands at 105, i.e. 4 nanoseconds versus 4 milliseconds. For scientific applications, which are tend to be write oriented, this gap can be worse, acting as a bottleneck towards the speed of application execution.

4.5.2 Computation overhead

Running a process on an idle processor vs. running it on a busy processing core widely affect the performance of application. This is called computation overhead where the process finds comparatively less computation time due to other running process eating the processing power of processor leaving a no room for application processes to run at its maximum speed. Computation overhead is a critical performance issue for scientific application which includes large amount of number crunching or to run computationally complex simulations in real time.

4.5.3 Process communication overhead

For parallel executing applications, inter-processes communication can be a significant fraction on the overall execution of the application and results in suboptimal performance in the execution. In PanCGP, process communication can be of three types including:

- Message passing
- Shared memory

4.5.3.1 Message passing

This method is most commonly used in distributed application following distributed memory architecture. In PanCGP, processes running on different machines use this method for synchronization by sending message to other process to interact.

4.5.3.2 Shared memory

This method is followed by parallel applications running on shared memory architecture. Processes that run on same computing machine follows this architecture for communication.

Chapter 5

Conclusion & Future Work

In this thesis, an optimized methodology for pan-genome analysis is proposed based on high performance computing . It implements divide and conquer approach using data decomposition technique to divide the input dataset into smaller sized chunks and run the whole pipeline on each chunk. Parallel processing techniques are used to maximize the performance and minimize the time cost of the pipeline. To ensure the nested parallelism on hybrid cluster architecture where each node is comprised of multi-core processors, MPJ Express [30] has been incorporated which underlying uses JavaMPI and java openmp(JOMP) to exploit the processing overhead on each core.

PanCGP pipeline performance has been evaluated on multiple parameters like time cost, scalability, efficiency and accuracy. The results has also been compared with the results of previously used CMG-biotools [26] pan-genome analysis pipeline which was taken as base case implementation. It is observed that minimizing the size of input data shorten the processing time of pipeline. And because each data chunk is processed independently so this results in no communication overhead required for processing the results. Since, the pipeline is divided into multiple steps and each step has its I/O integrated with other steps and results calculated in a step are fed as input to the other, so there is synchronization mechanism involved for all the thread to complete one particular step and then begin with the next step to ensure that results computation of previous step is accomplished. The PanCGP pipeline is implemented on cluster machine to compare the analytical and experimental results. Analytical comparisons and the experimental results demonstrate that performance of PanCGP pipeline is much better than the one provided in CMG bitools [26] and it has much better scalability factor as compared to the CMG-biotools [26].

The modular nature of PanCGP pipeline makes it easier to customize it by replacing the tools and software being used in the pipeline for the analysis purpose. As well as extending the pipeline by adding new tools makes it more flexible towards customization, future enhancements and optimizations. On the other hand, flexibility comes at some cost. Modular nature of PanCGP results in the cost of synchronization barriers on all modules I/O to ensure that results computation of last module has been accomplished before feeding them to the next module as input.

5.1 Conclusion

In this thesis, we presented an enhanced and optimized PanCGP pipeline tool based on high performance computing. Parallel programming techniques are incorporated to enhance the processing power and scalability of pipeline. For testing and evaluation purposes, we compared the benchmarks of PanCGP with previously used CMG-biotools [26]. A sample dataset of 38 strains of *H. Pylori* was fed as input to compute the results. From analytical and experimental benchmark results, it has been observed that our proposed PanCGP pipeline is more scalable and computationally more efficient than the previous. For sequence comparison, BLASTALL is used with TIGRCUT utility to satisfy the 50/50 sequence matching rule. The results has been compared with the previously used CMG-biotools pan-genome analysis pipeline and best result accuracy has been achieved.

5.2 Future work

In last decade, a number of software and tools has been developed for genetic analysis, having its own functionalities, features and restrictions, finally making a room for future enhancements. For future work, we recommend the porting of pipeline to heterogeneous architectures for further optimization and efficiency.

Heterogeneous architectures include:

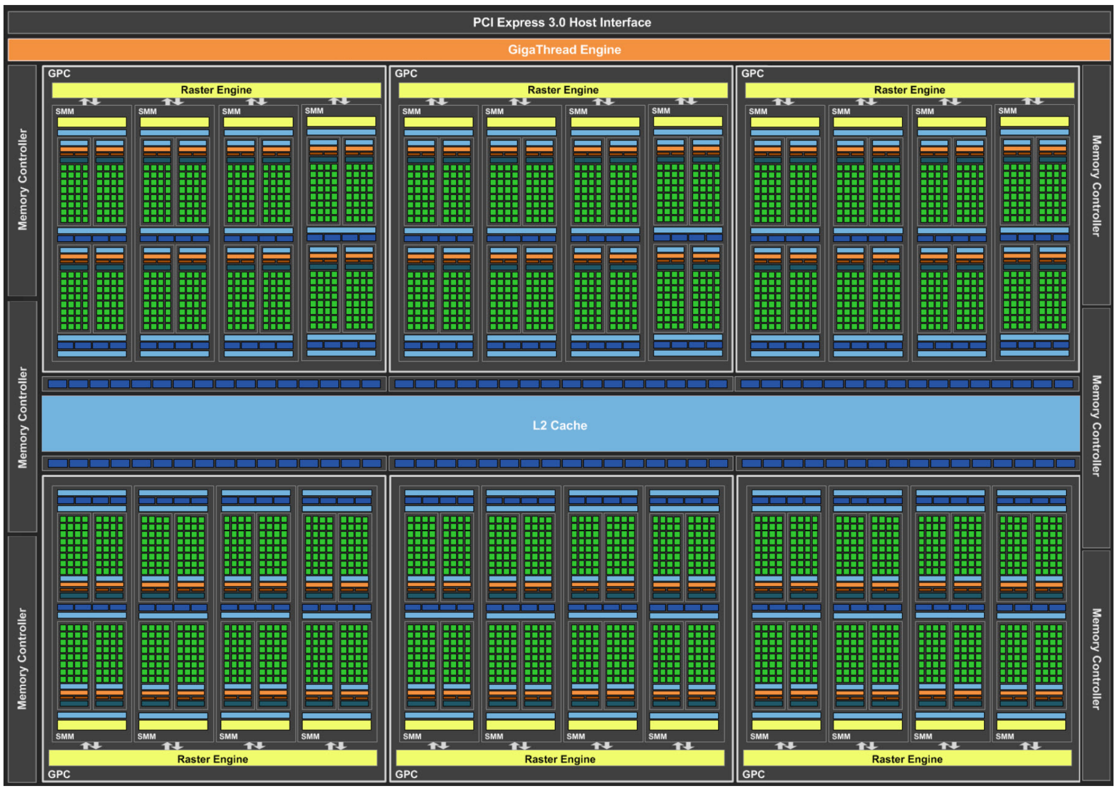

Figure 5.1: NVidia Gpu Maxwell GM206 Chip Architecture [33]

- GPU architecture
- Embedded architecture

5.2.1 GPU architecture

Porting the PanCGP pipeline to latest GPU architectures will greater optimize the overall performance of pipline. GPUs has revolutionized the world of high performance computing by enhancing the computation power of Super Computers and minimizing the energy consumption required to perform that computations. After comparing the architecture of modern CPUs and GPUs, it has been observed that modern GPUs possess high computation power with comparatively less energy consumption. They are comprised of thousands of computation cores able to process a large amount of data as compared to modern CPUs available in the market. Below are the figures 5.1 and 5.2 showing the architecture of GM206 gpu chip and intel Xeon 7500 series processor chip respectively.

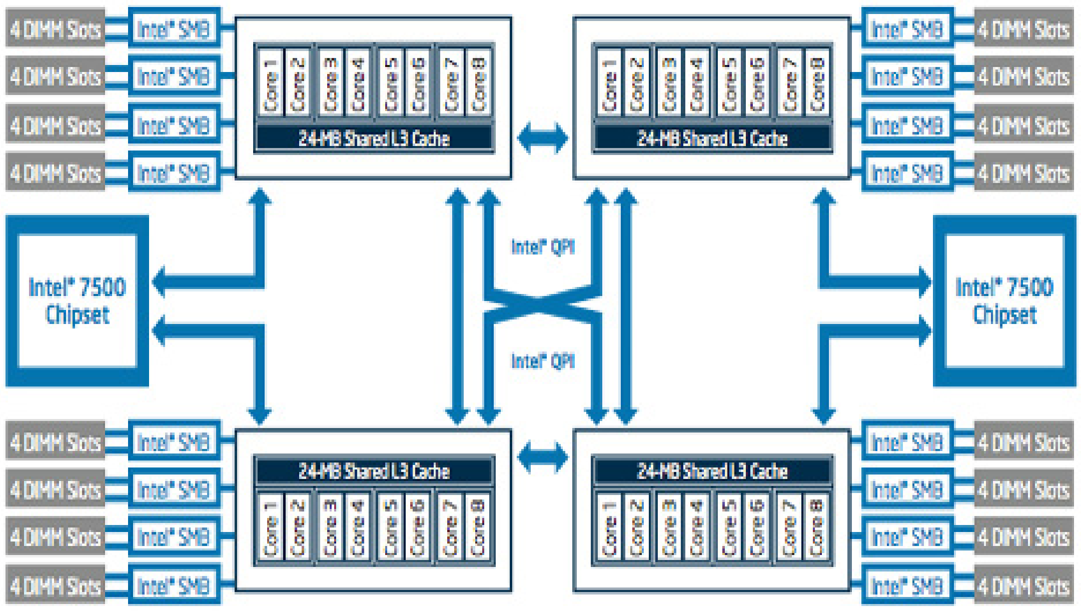

Figure 5.2: Intel IT silicon design xeon processor 7500 series chip

5.2.2 Embedded architecture

Another perspective of future optimization is porting the pipeline to embedded architectures including FPGAs and embedded Supercomputing architectures which will greatly enhance the overall performance and efficiency of pipeline and minimize the time cost up to minimum. FPGAs has multiple advantages. They operate on hardware implemented algorithms making it much faster than any other software based implementation. Hardware implementation makes the FPGAs more deterministic and their latency are order of magnitude of nanoseconds with far less power consumption as compare to both the cpus and gpus.

Chapter 6

Installation & user guide

6.1 Installation

Download and install Ubuntu, Ubuntu 12.04 or latter but Ubuntu 12.04 desktop version is recommended because all tools are tested at this version. You may also use latter version but you should take care of compatibilities of all relevant tools.

1. Open terminal using ALT+CTRL+T and run following commands one by one

   - sudo apt-get update
   - sudo apt-get install gcc build-essential
   - sudo apt-get install default-jre
   - sudo apt-get install default-jdk
   - sudo apt-get install -y autotools-dev g++ build-essential openmpi1.5-bin openmpi1.5-doc libopenmpi1.5-dev
   - sudo apt-get autoremove
   - sudo apt-get install perl
   - sudo apt-get install bioperl

   (At any step, if system prompts for any input, please press *Enter* and use default option.)

2. Download *mpjexpress* library from here. (Current latest is 0.44)

3. Once downloaded, extract it to a directory (suppose that you have downloaded the it and extracted in your home directory then the path will be */home/<username>/mpj/mpj-v_044*)

4. Now we need to set environment variables for mpj express. These variables can be added in .bashrc file. The variables need to be added in start of file otherwise may cause problem.

    - Open .bashrc: *vi .bashrc*
    - For java path: *export JAVA_HOME=/usr/java/latest*
    - For mpj express path: *export MPJ_HOME=/home/<username>/mpj /mpj-v_044*
    - Add to path variable: *export PATH=$MPJ_HOME/bin:$JAVA_HOME /bin:$PATH*
    - Add library paths: *export LD_LIRARY_PATH=$MPJ_HOME/lib: $LD_LIBRARY_PATH*

5. Download the PanCGP package from here:

    - Extract it to your home directory, the path may be: */home/<username> /PanCGP*
    - Now, we need to add path to environment variables
    - For PanCGP path, open .bashrc: *vi .bashrc*
    - For adding path: *export PANCGP=/home/<username>/PanCGP /depends/*
    - Add blast path: *export PANCGPBLAST=/home/<username> /PanCGP/depends/blast/bin*
    - Add to path variable: *export PATH=$PANCGP:$PANCGPBLAST: $PATH*
    - Set permissions of files in path to allow execution (with extension or without extension): *sudo chmod +x **
      *sudo chmod +x *.**

Now, PanCGP has been installed and ready to use for analysis purposes.

6.2 User guide

PanCGP can be run in two modes: *Single-node* and *multi-node* cluster modes.

6.2.1 Single node

Assuming that user has successfully carried out the installation. To execute the PanCGP pipeline:

1. As explained in installation, after extracting the PanCGP pipeline the path may be: */home/<username>/PanCGP*
2. Execute: *cd /home/<username>/PanCGP*
3. Execute: *mpjrun.sh -np <total no. of threads to run> -jar PanCGP.jar <Path to input folder containing FASTA files>*

6.2.2 Multi-node

This section list down the steps to run the PanCGP pipeline to run in a multi-node or cluster architecture.

1. Step 1 and 2 from above section are same.
2. To run PanCGP on multiple nodes, a $machine\_file$ is required, a text file listing all the IP or aliases of machines or nodes (which are fully qualified name of nodes in reality) on which pipeline is intended to execute.
   Suppose that there are three machines: machine1, machine2, machine3. In $machine\_file$, these will be listed as:

   ```
   machine1
   machine2
   machine3
   ```

3. Execute: *mpjboot <name of machine_file>*
   This will start daemon processes on machines or nodes listed in the file. These daemon processes will host the parallel processes launched by executing PanCGP command.

4. Execute: *mpjrun.sh -np <total no. of threads to run> -dev niodev -jar PanCGP.jar <path to input folder containing FASTA files>*

5. Once the pipeline has completed execution, execute: *mpjhalt <name of machine_file>*
   This command will stop the daemon processes running on machines or nodes listed in the file. There daemons are not necessary to stop after every execution. Once an execution is complete, they are ready to host another execution of pipeline.

Bibliography

[1] Dennis A Benson, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, and David L Wheeler. Genbank. *Nucleic acids research*, 36(suppl 1):D25–D30, 2008.

[2] Axel Bernal, Uy Ear, and Nikos Kyrpides. Genomes online database (gold): a monitor of genome projects world-wide. *Nucleic Acids Research*, 29(1): 126–127, 2001.

[3] RA Welch, V Burland, GIII Plunkett, P Redford, P Roesch, D Rasko, EL Buckles, S-R Liou, A Boutin, J Hackett, et al. Extensive mosaic structure revealed by the complete genome sequence of uropathogenic escherichia coli. *Proceedings of the National Academy of Sciences*, 99(26):17020–17024, 2002.

[4] Hervé Tettelin, Vega Masignani, Michael J Cieslewicz, Claudio Donati, Duccio Medini, Naomi L Ward, Samuel V Angiuoli, Jonathan Crabtree, Amanda L Jones, A Scott Durkin, et al. Genome analysis of multiple pathogenic isolates of streptococcus agalactiae: implications for the microbial pan-genome. *Proceedings of the National Academy of Sciences of the United States of America*, 102(39):13950–13955, 2005.

[5] Hervé Tettelin, David Riley, Ciro Cattuto, and Duccio Medini. Comparative genomics: the bacterial pan-genome. *Current opinion in microbiology*, 11(5): 472–477, 2008.

[6] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[7] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast:

a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[8] Jung D Oh, Helene Kling-Bäckhed, Marios Giannakis, Jian Xu, Robert S Fulton, Lucinda A Fulton, Holland S Cordum, Chunyan Wang, Glendoria Elliott, Jennifer Edwards, et al. The complete genome sequence of a chronic atrophic gastritis helicobacter pylori strain: evolution during disease progression. *Proceedings of the National Academy of Sciences*, 103(26):9999–10004, 2006.

[9] Michal Wozniak, Limsoon Wong, and Jerzy Tiuryn. Camber: an approach to support comparative analysis of multiple bacterial strains. *BMC genomics*, 12(Suppl 2):S6, 2011.

[10] Chad Laing, Cody Buchanan, Eduardo N Taboada, Yongxiang Zhang, Andrew Kropinski, Andre Villegas, James E Thomas, and Victor PJ Gannon. Pan-genome sequence analysis using panseq: an online tool for the rapid analysis of core and accessory genomic regions. *BMC bioinformatics*, 11(1):461, 2010.

[11] Mitchell J Brittnacher, Christine Fong, HS Hayden, MA Jacobs, Matthew Radey, and Laurence Rohmer. Pgat: a multistrain analysis resource for microbial genomes. *Bioinformatics*, 27(17):2429–2430, 2011.

[12] Jumamurat R Bayjanov, Roland J Siezen, and Sacha AFT van Hijum. Pancghweb: a web tool for genotype calling in pangenome cgh data. *Bioinformatics*, 26(9):1256–1257, 2010.

[13] Yongbing Zhao, Jiayan Wu, Junhui Yang, Shixiang Sun, Jingfa Xiao, and Jun Yu. Pgap: pan-genomes analysis pipeline. *Bioinformatics*, 28(3):416–418, 2012.

[14] Matthew N Benedict, James R Henriksen, William W Metcalf, Rachel J Whitaker, and Nathan D Price. Itep: An integrated toolkit for exploration of microbial pan-genomes. *BMC genomics*, 15(1):8, 2014.

[15] Todd J Treangen, Brian D Ondov, Sergey Koren, and Adam M Phillippy. The harvest suite for rapid core-genome alignment and visualization of thousands of intraspecific microbial genomes. *Genome Biol*, 15(524):2, 2014.

[16] Bruno Contreras-Moreira and Pablo Vinuesa. Get_homologues, a versatile software package for scalable and robust microbial pangenome analysis. *Applied and environmental microbiology*, 79(24):7696–7701, 2013.

[17] Corinna Ernst and Sven Rahmann. Pancake: A data structure for pangenomes. In *GCB*, pages 35–45, 2013.

[18] Yongbing Zhao, Xinmiao Jia, Junhui Yang, Yunchao Ling, Zhang Zhang, Jun Yu, Jiayan Wu, and Jingfa Xiao. Pangp: a tool for quickly analyzing bacterial pan-genome profile. *Bioinformatics*, 30(9):1297–1299, 2014.

[19] AR Santos, E Barbosa, K Fiaux, M Zurita-Turk, V Chaitankar, B Kamapantula, A Abdelzaher, P Ghosh, S Tiwari, N Barve, et al. Pannotator: an automated tool for annotation of pan-genomes. *Genet. Mol. Res*, 12:2982–2989, 2013.

[20] Dennis A Benson, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, Barbara A Rapp, and David L Wheeler. Genbank. *Nucleic acids research*, 28 (1):15–18, 2000.

[21] UniProt Consortium et al. Reorganizing the protein space at the universal protein resource (uniprot). *Nucleic acids research*, page gkr981, 2011.

[22] Brigitte Boeckmann, Amos Bairoch, Rolf Apweiler, Marie-Claude Blatter, Anne Estreicher, Elisabeth Gasteiger, Maria J Martin, Karine Michoud, Claire O'Donovan, Isabelle Phan, et al. The swiss-prot protein knowledgebase and its supplement trembl in 2003. *Nucleic acids research*, 31(1):365–370, 2003.

[23] Alex Bateman, Lachlan Coin, Richard Durbin, Robert D Finn, Volker Hollich, Sam Griffiths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik LL Sonnhammer, et al. The pfam protein families database. *Nucleic acids research*, 32(suppl 1):D138–D141, 2004.

[24] Paul Flicek, M Ridwan Amode, Daniel Barrell, Kathryn Beal, Simon Brent, Denise Carvalho-Silva, Peter Clapham, Guy Coates, Susan Fairley, Stephen Fitzgerald, et al. Ensembl 2012. *Nucleic acids research*, page gkr991, 2011.

[25] Ji He, Xinbin Dai, and Xuechun Zhao. Associative artificial neural network for discovery of highly correlated gene groups based on gene ontology and gene

expression. In *Computational Intelligence and Bioinformatics and Computational Biology, 2007. CIBCB'07. IEEE Symposium on*, pages 17–24. IEEE, 2007.

[26] Tammi Vesth, Karin Lagesen, Oncel Acar, and David Ussery. Cmg-biotools, a free workbench for basic comparative microbial genomics. *PloS one*, 8(4): e60120, 2013.

[27] Doug Hyatt, Gwo-Liang Chen, Philip F LoCascio, Miriam L Land, Frank W Larimer, and Loren J Hauser. Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC bioinformatics*, 11(1):119, 2010.

[28] Gene M Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485. ACM, 1967.

[29] Michael J Flynn. Some computer organizations and their effectiveness. *Computers, IEEE Transactions on*, 100(9):948–960, 1972.

[30] Aamir Shafi, Bryan Carpenter, and Mark Baker. Nested parallelism for multicore hpc systems using java. *Journal of Parallel and Distributed Computing*, 69(6):532–545, 2009.

[31] Jean-F Tomb, Owen White, Anthony R Kerlavage, Rebecca A Clayton, Granger G Sutton, Robert D Fleischmann, Karen A Ketchum, Hans Peter Klenk, Steven Gill, Brian A Dougherty, et al. The complete genome sequence of the gastric pathogen helicobacter pylori. *Nature*, 388(6642):539–547, 1997.

[32] Johannes G Kusters, Arnoud HM van Vliet, and Ernst J Kuipers. Pathogenesis of helicobacter pylori infection. *Clinical microbiology reviews*, 19(3): 449–490, 2006.

[33] C Nvidia. Nvidias next generation cuda compute architecture: Kepler gk110. Technical report, Technical report, 2012.[28] http://www. top500. org, 2012.