靜宜大學資訊工程學系
碩士論文

Department of Computer Science and Information,

Providence University

Master Thesis

MapReduce 架構下的單體型區塊切割與

單核甘酸多型體之標籤選擇

Haplotype Block Partitioning and TagSNP Selection

with MapReduce Framework

研究生：滑冠傑
Graduate Student: Guan-Jie Hua

指導教授：林耀鈴 博士
洪哲倫 博士
Advisor: Yaw-Ling Lin, Ph.D.
Che-Lun Hung, Ph.D.

中華民國一百零二年七月

July, 2013

MapReduce 架構下的單體型區塊切割與單核甘酸多型體之標籤選擇

Haplotype Block Partitioning and TagSNP Selection with MapReduce Framework

研究生：滑冠傑　　Student：Guan-Jie Hua

指導教授：林耀鈴　博士　　Advisor：Yaw-Ling Lin, Ph. D.

洪哲倫　博士　　Che-Lun Hung, Ph. D.

靜宜大學

資 訊 工 程 學 系

碩 士 論 文

A Thesis

Submitted to Department of Computer Science and Information Management

College of Computing and Informatics

Providence University

in partial Fulfillment of the Requirements

for the Degree of Master

in

Computer Science and Information Management

July 2013

Shalu,Taichung, Taiwan, Republic of China

中華民國一百零二年七月

靜 宜 大 學

論 文 口 試 委 員 會 審 定 書

本校　資訊工程　學系 碩士班　滑冠傑　君

所提論文：

（中）MapReduce 架構下的單體型區塊切割與單核甘酸多型體之標籤選擇

（英）Haplotype Block Partitioning and TagSNP Selection with MapReduce framework

合於碩士資格水準、業經本委員會評審通過。

口試委員：訂青蓉

指導教授：林朝欽

沈哲倫

系主任：翁慶昭

中　華　民　國　1　0　2　年　7　月　8　日

誌　　謝

在逐漸摸索資訊領域的懵懂過程，對於很多課程和研究，我很感謝我的指導教授林耀鈴博士，循循善誘，耐心的引導我的學習，對我有很大的幫助；教授重視的更是研究的態度與想法，印象深刻的話如：對小事看輕，對大事懷疑，那將什麼也做不好；或是哥倫布的蛋的故事，來刺激我如何去開創自己的思考模式。

我的共同指導教授，洪哲倫博士對於雲端資訊非常擅長，在很多地方為我指點迷津。我很感謝他鼓勵我嘗試更多的發展，與我討論未來的方向。

我也很感謝口試委員許芳榮教授在口試時，給予很多指教與建議，讓我能夠對於做研究的學問與方法上能夠改進。

我也要感謝實驗室的成員，謝承恩以及徐培昇一同討論和參與研究，給予不少幫助。

最後感謝我的家人，默默的支持著我，讓我可以專心完成學業。

在計算機科學的領域，我深深覺得還有許多還需要去學習研究的地方，未來我也會繼續研究精進這塊領域。

滑冠傑　　謹誌

民國一百零二年七月

MapReduce 架構下的單體型區塊切割與單核甘酸多型體之標籤選擇

學生：滑冠傑　　指導教授：林耀鈴 博士
洪哲倫 博士

靜宜大學資訊工程學系碩士班

摘　要

單核苷酸多型體在各種分析應用，包括醫療診斷和藥物設計中扮演了重要角色。它們包含了最高分辨率的基因指紋識別來關聯疾病與人類特徵。單倍體，由單核苷酸多型體組成，因連鎖遺傳變異，鄰近區段常常一併被繼承下來。最近，遺傳學研究表明，特定的單倍型區塊誘使出只有幾種常見的單體型，在主要的人類族群中。單倍型塊的討論基於疾病基因的關聯與定位方法上有重大的影響。

我們提出的方法，調查了許多以前的文獻中相關的一些有效的組合算法，去根據不同的多樣性算式，選擇感興趣的單倍型塊。然而，這些方法計算相當耗時。本論文採用的方法，使用 MapReduce 去平行化和管理其程式執行。實驗結果表明，原始的單執行序程式經過了 map/reduce 平行，將現有的 HapMap 的資料庫獲得的數據做分析，計算的效率將以所使用的處理器數目成比例的成長，可以有效地提高計算效能。

關鍵字：演算法，SNP haplotype，tag SNP，haplotype 區塊分割，Hadoop，MapReduce。

Haplotype Block Partitioning and TagSNP Selection with MapReduce Framework

Student：Guan-Jie Hua　　Advisors：Dr. Yaw-Ling Lin

Dr. Che-Lun Hung

Department of Computer Science and Information Management
Providence University

ABSTRACT

SNPs play important roles for various analysis applications including medical diagnostic and drug design. They contain the highest-resolution genetic fingerprint for identifying disease associations and human features. Haplotype, is composed of SNPs, region of linked genetic variants that are neighboring usually inherited together. Recently, genetics researches show that SNPs within certain haplotype blocks induce only a few distinct common haplotypes in the majority of the population. The discussion of haplotype block has serious implications of method with association-based for the disease genes mapping.

We proposed the method in investigating several efficient combinatorial algorithms related to selecting interesting haplotype blocks under different diversity functions that generalizes many previous results in the literatures. However, the proposed method is computation-consuming. This thesis adopts approach using the MapReduce paradigm to parallelize tools and manage their execution. The experiment shows that the map/reduce-paralleled from the original sequential combinatorial algorithm performs well on the real-world data

obtained in from the HapMap data set; the computation efficiency can be effectively improved proportional to the number of processors being used.

Keywords: algorithm, SNP haplotype, tag SNP, haplotype block partition, Hadoop, MapReduce.

Contents

Figures

Chapter 1

Introduction

1.1 SNPs to Haplotypes

A SNP(Single Nucleotide Polymorphism), is a small genetic variation, that occur within human's DNA sequence. DNA sequence is composed by the four nucleotide letters A (adenine), C (cytosine), T (thymine), and G (guanine). Genetic variation is a single nucleotide in the genome (or other shared sequence) differs between members of a biological species or paired chromosomes in a human. However, a SNP is a single nucleotide changing to one of the other three nucleotides, most SNPs are transitive substitution between purines (A,G) or pyrimidines (C,T). For example, in Figure 1, one may has a C at a particular site in the chromosome, the another is different with T at the same position. Each form of a SNP is called an allele. Each person has one pair of all chromosomes except the sex chromosomes. The set of allele pairs that a person has is called a genotype. For this SNP, a person could have three genotypes, CC, CT, or TT. The term "genotype" can refer to the SNP alleles that a person has at a SNP in particular site, or for a SNPs sequence across the genome. A method that discovers what genotype a person has is called genotyping. SNP occurs about per 2,000 base pairs in human genome.

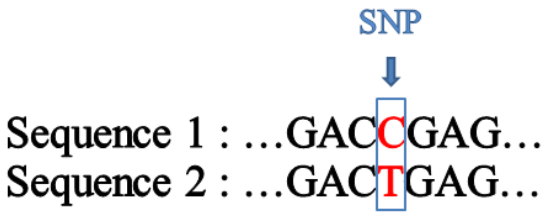

Figure 1: SNP in human genome.

Because most SNPs just have two alleles at each site, the one with higher frequency is called major allele, and the other one is called minor allele. About 1.4 million SNPs exist in human populations, where the minor allele has a frequency of at least 1%. Because only about 3 to 5 percent of a person's DNA sequence codes for the production of proteins, most SNPs are found outside of coding sequences (exon). SNPs found within a exon are of particular interest to researchers because they are more likely to alter the biological function of a protein. A SNP in a coding region may have two different effects on the resulting protein. The one is termed as synonymous mutation if the substitution causes no amino acid change to the protein it produces. This is also called a silent mutation. On the other hand, the non-synonymous mutation is the substitution that causing an alteration of the encoded amino acid. The non-synonymous mutations will change the protein by causing codon changed, referred to as missense mutation, or effect in a misplaced termination codon, termed as nonsense mutation. One half of all coding sequence SNPs result in non-synonymous codon changes.

Alleles of SNPs that are close together tend to be inherited together. A haplotype refers to a set of SNPs found to be statistically associated on a single chromosome. The haplotypes in the human genome have been produced by the molecular mechanisms of zoogamy and by the history of our species. The chromosomes in human cells appear in pairs with the exception of the germ cells. One member of each chromosome pair is inherited form a person's father and

the other member of the pair is inherited from that person's mother. But each haploid chromosome does not pass from each generation to the next as identical copy. Rather, when sperms and ova are being formed, the chromosome pairs suffer a process known as crossover and recombination shown in Figure 2. The members of each chromosome pair will interchange some pieces and result in a hybrid chromosome containing pieces from both members of a chromosome pair. After the process of meiosis, each sperm and ovum will contain a hybrid chromosome. A pair of new formed chromosome will be passed to the next generation by the process of insemination.

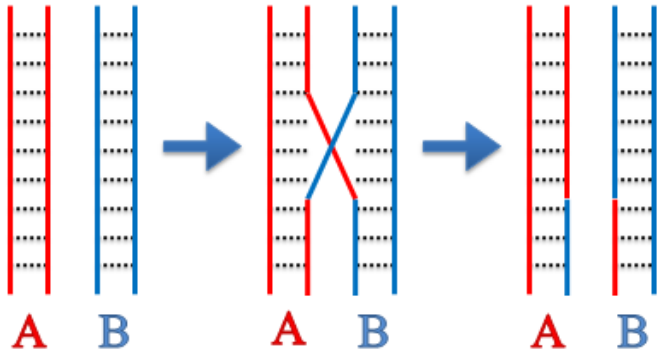

Figure 2: The crossover and recombination events of chromosome.

Over the same course of many generations, segments of the ancestral chromosomes in an interbreeding population are shuffled through repeated recombination events. Some of the segments of the ancestral chromosomes appear as regions of genome sequences that are shared by multiple individuals. These segments are regions of chromosomes that have not been broken up by recombination, and they are separated by places where recombination has occurred. These segments are the haplotypes that enable geneticists to search for genes involved in diseases and other medically important traits. The most

information of SNPs and haplotypes in collected by the International HapMap Project [1].

Figure 3 illustrates an example of four haploid chromosome to produce four haplotypes. Because each SNP has two alleles, we can use 0 to represent the major allele and use 1 to represent the minor allele. Thus, these data can be arranged and form an $m \times n$ haplotype matrix; m is the number of haplotypes, and n is the number of SNPs.

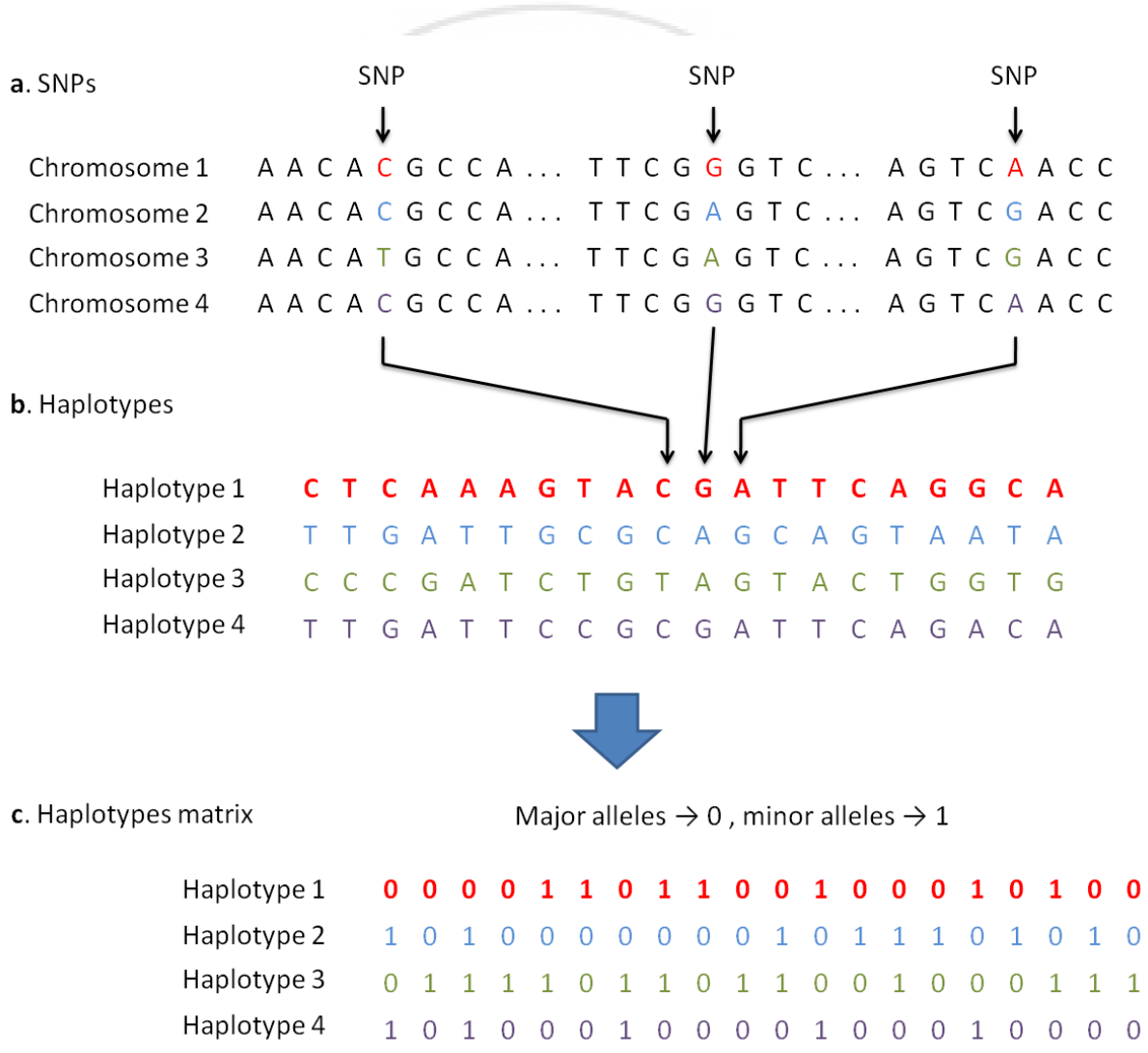

Figure 3: Haplotypes and haplotype matrix.

1.2 Motivation and Purpose

Although the DNA sequences of any two unrelated people are the same at about 99.9% [2], the remaining 0.1% is important because it contains the genetic variations that influence how people differ in their risk of disease or their response to drugs. Discovering the DNA sequence variations that contribute to common disease risk offers one of the best opportunities for understanding the complex causes of disease in humans.

SNPs are the most common form of DNA sequence variation. They are useful polymorphic markers to investigate genes susceptible to diseases or those related to drug responsiveness. Furthermore, a small subset of SNPs directly influences the quality or quantity of the gene product, and increase a risk to certain diseases and to severe side effect by drugs. Through a discovery of a large number of SNPs, many research have contributed to identification of disease-related genes and also to establish a diagnostic method to avoid drug side-effect.

Linkage disequilibrium (LD) is a term used in the study of population genetics for the non-random association of alleles at two or more loci, not necessarily on the same chromosome. Linkage disequilibrium describes a situation where some combinations of alleles or genetic markers occur more or less frequently in a population than would be expected from a random formation of haplotypes from alleles based on their frequencies.

In recent years, the patterns of linkage disequilibrium observed in the haplotypes of human population reveal a block-like structure [3, 4, 5, 6]. The entire chromosome can be partitioned into high LD regions interspersed by low LD regions. The high LD regions are usually called haplotype block and the low

LD ones are referred to as recombination hotspots [7]. Within a haplotype block, there is little or no recombination that occurs and the SNPs are highly correlated. There are only a few common haplotypes, that account for most of the variation from person to person, in a haplotype blocks.

Due to the low diversity in each haplotype block, SNPs, haplotypes, or disease genes in the same block are associative. With this knowledge, it is thought that the identification of a few alleles of a haplotype block can unambiguously identify all other polymorphic sites in this region. Such information is very valuable for investigating the genetics behind common diseases [8].

"Tagging" SNPs are aimed at characterizing candidate genes avoiding redundancies in genotyping. Most of the tagging SNP selection strategies are haplotype based. The aim is to identify a minimal subset of SNPs that can characterize the most common haplotypes [3, 9]. These SNPs are referred to as haplotype tagging SNPs, also termed as tag SNPs, which are markers that capture most of the haplotypes in a region of linkage disequilibrium. For example, in Figure 4, it just require three tag SNPs to account for the information about the haplotype block. A number of articles show that it is possible to retain much of the information of haplotypes by retaining only a reduced subset of markers. Haplotype blocks refer to sites of closely located SNPs which are inherited in blocks. Regions corresponding to blocks have a few common haplotypes which account for a large proportion of chromosomes. Identification of haplotype blocks is a way of examining the extent of LD in the genome, which generally provides useful information for the planning of association studies.

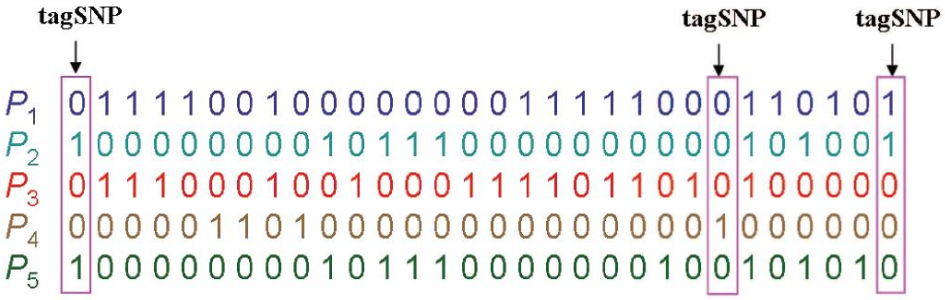

Figure 4: The information about haplotypes in a block can be obtained by using a few tag SNPs

The characteristics of haplotype blocks and tag SNPs are very important and useful for medicine and therapy. We can analyze the relation between certain haplotype pattern and disease gene if one chromosome range contains disease gene but no recombination event occurred. So if someone, such as a newborn baby, is doubted with certain of inherited diseases, we can identify the haplotype pattern by checking tag SNPs or haplotypes to infer whether the individual with inherited disease or not, instead of scanning whole genome sequence. Studying on SNPs and haplotype blocks not only decreases the cost for detecting inherited diseases but also has many contributions for classifying the race of humans and researching on species evolution.

SNP haplotype patterns and disease genes in the same blocks are associative [5, 10], and therefore we can analyze the relation between certain of haplotype patterns and disease gene if a chromosome region contains disease genes but no recombination occurred. Tag SNPs can capture most of the haplotype diversity in the blocks, and therefore could potentially capture most of the information for association between a trait and the SNP marker loci. We can figure out the diversity and features of each haplotype block easily and economically by using

tag SNPs. For these reasons, we want to find the longest haplotype blocks such that the number of tag SNPs is minimized. In this thesis we get some haplotype samples from several public data and apply the recombination mechanism in inherited process to develop a suitable observation and analysis to these haplotype samples. Following we propose some dynamic programming algorithms to partition those haplotype sample into haplotype blocks according to some measurement functions, such that all SNPs in the same haplotype block can keep important information about evolution and inheritance. We also develop algorithms to select tag SNPs for each haplotype block. Using these algorithms we can obtain information about human genome as more as possible by checking only a few tag SNPs. These characteristics will help biologists and doctors for determining or analyzing some subjects related to the inherited diseases.

1.3 Hadoop and MapReduce

Hadoop [11] is a software framework intended to support data-intensive distributed applications. It is able to process petabytes of data with thousands of nodes. Hadoop supports MapReduce programming model [12] for writing applications that process large data set in parallel on Cloud Computing environment. The advantage of MapReduce is that it allows for distributed computing of the map and reduce operations. Each map operation is independent of the other and all maps can perform the tasks in parallel. In practice, the number of the map is limited by the data source and/or the number of CPUs near that data. Similarly, a set of reducers can perform the reduce operations. All outputs of the map operations which share the same key are presented to the

same reducer, at the same time. In addition, one of the important benefits to use Hadoop to develop the applications is due to its high degree of fault tolerance. Even when running jobs on a large cluster where individual nodes or network components may experience high rates of failure, Hadoop can guide jobs toward a successful completion.

Many applications of bioinformatics are often computation-consuming; sometimes it needs weeks or months to complete the jobs. The traditional parallel models, such as MPI, OpenMP and Multi-thread, are not suitable to such applications, where a fault occurred in some nodes leads the entire application into total failure. In these sitrations, the Hadoop platform are considered as a much better solution for these real-world applications. Recently, Hadoop has been applied in various domains in bioinformatics [13,14].

Chapter 2

Related Works

2.1 Different Measurement for Haplotype partitioning

The result of block partition and the meaning of each haplotype block may be different by using different measuring methods. Our ultimate goal is to select haplotype block designations that best capture the structure of human genome within the genotype data.

Unfortunately, a consensus definition for haplotype blocks based on the LD structure has not been established thus far. However, a range of operational definitions has been used to identify haplotype-block structures, including diversity-based [3, 9, 15, 16], LD-based [4,18], recombination-based [19, 20], and information-complexity-
based [16, 17, 18] methods.

For a diversity-based test, methods can be classified into two categories: those that divide strings of SNPs into blocks on the basis of the decay of LD across block boundaries and those that delineate blocks on the basis of some haplotype-diversity measure within the blocks. Patil et al. [3] defined a haplotype block as a region where at least 80% of observed haplotypes within a block are represented more than once. They used a greedy algorithm to partition human Chromosome 21 into haplotype blocks in a sample of 20 re-sequenced chromosomes. Their algorithm considers all blocks of consecutive SNPs of one SNP or larger, and eliminates overlapping block by choosing the block with the

maximum ratio of SNPs in the block to the number of tag SNPs required to discriminate all haplotypes represented in the block. The process was repeated until the entire length of the chromosome was partitioned into haplotype blocks. Subsequently, Zhang et al. [9, 16] proposed a dynamic programming algorithm, and defined block boundaries in a way that minimizes the number of tag SNPs that are required to identify all the haplotype in a block. Under the same criteria, the algorithm can get the better results.

For a LD-based test, Gabriel et al. [4] define haplotype blocks to be a region in which a small proportion of marker pairs show evidence for historical recombination. These criteria are suggestively modified by Wall and Pritchard [18] for handling haplotype data instead of unphased genotype data. They computed confidence bounds of the value of *D*' , a standard measurement of LD [21, 22], and defined pairs of SNPs to be in strong LD (little evidence of recombination) if the one-sided 95% *D*' confidence bound is between 0.7 and 0.98. Blocks are partitioned according to whether the upper and lower confidence limits on estimates of pairwise D' measure fall within the threshold values.

The recombination-based method uses the four-gamete test of Hudson and Kaplan [19] suggested by Wang et al. [20]; they define haplotype blocks as apparently recombination-free regions under the infinite-sites assumption.

The information complexity based method has been proposed by Anderson et al. [23]; they develop the method by formalizing the task of finding block boundaries as a problem in statistical-model selection, where they apply the minimum description length (MDL) criterion to select the block designations that best capture the structure within the data. The MDL criterion is an application of information theory to statistical-model selection. The description

length of a data set is a penalized negative log-likelihood, which is a function of the number and position of block boundaries. The best set of block boundaries, by the MDL criterion, is the set that achieves the shortest description length for the data.

Koivisto et al. [24] report a method that uses an underlying probability model, haplotypes within blocks are clustered using k-means clustering, and the description length depends on the number of clusters within a block and how closely haplotypes within blocks cluster together. Their method is thus based on a measure of within-block haplotype diversity. Greenspan and Geiger [25] also present an MDL-based haplotype block partitioning method that has the attractive feature in that it accepts both phased haplotype data and unphased genotype data.

By using appropriate diversity functions, the block selection problem can be viewed as finding a segmentation of given haplotype matrix such that the diversities of chosen blocks satisfy certain value constraint.

2.2 Common Haplotype

Patil et al. [3] defined a haplotype block as a region where at least 80% of observed haplotypes within a block must be common haplotype. One of the major objectives of Patil et al. is to characterize the common haplotypes. To define the common haplotypes in a block, we need to first introduce the concept of ambiguous and unambiguous haplotypes as in Patil et al. when missing data are present.

Two haplotypes are said to be compatible if the alleles are identical at all loci for which there are no missing data; otherwise the two haplotypes are said to be

incompatible. As in Patil and Zhang [9], we define the ambiguous haplotypes as those haplotypes compatible with at least two haplotypes that are themselves incompatible. It should be noted that when there are no missing data, all of the haplotypes are unambiguous. As in Patil [3] and Zhang [9], we define the common haplotypes as those haplotypes that are represented more than once in a block. The haplotypes are called singleton if they are not compatible with any others.

For example, consider six haplotypes h_1 = (1, 1, 0, 1, 0), h_2 = (1, 3, 3, 1, 0), h_3 =
(3, 3, 0, 1, 3), h_4 = (1, 0, 1, 0, 1), h_5 = (0, 0, 0, 3, 1) and h_6 = (0, 0, 3, 1, 1). Here 0, 1 and 3 denote the major allele, minor allele and missing data, respectively. Haplotype h_1 is compatible with h_2 because the alleles are the same for the two haplotypes at the loci with no missing data, by the same reason haplotype h_5 and h_6 are compatible. In this case, haplotype h_1, h_2, h_5 and h_6 are common haplotype, on the other hand haplotype h3 is ambiguous because it is compatible with haplotype h_1 and h_5, but h_1 and h_5 are not compatible with each other. Here haplotype h_4 is a singleton.

Patil et al. [3] require that at least ρ = 80% of the unambiguous haplotypes appear more than once. The ρ is also referred to as the coverage of common haplotypes in a block. Ambiguous haplotypes are not included in calculating percent coverage. Therefore, the coverage of block A can be mathematically formulated as a form of diversity:

$$\delta s(A) = 1 - \frac{C}{U} = \frac{S}{U}$$

Here U denotes the number of unambiguous haplotypes, C denotes the number of common haplotypes, and S denotes the number of singleton haplotypes. In other words, Patil et al. require that $\delta_S(A) \leq 20\%$.

2.3 Monotonic Diversity

The results of block partition and the meaning of each haplotype block may be different by using different measuring formulas. Given an m × n haplotype matrix A, a block A(i, j) (i, j are the block boundaries) of matrix A is viewed as m haplotype strings; they are partitioned into groups by merging identical haplotype strings into the same group. The probability pi of each haplotype pattern s_i, is defined accordingly such that $\sum p_i = 1$. As an example, Li [26] proposes a diversity formula defined by

$$\delta_D(S) = 1 - \sum_{s_i \in S} p_i^2$$

Note that $\delta_D(S)$ is the probability that two haplotype strings chosen at random from S are different from each other. According to the definition, we can classify each haplotype submatrix as a sample space (multiset).

Haplotype blocks are the genome regions with high LD, thus it imply that no matter what kinds of haplotype block definition we used, the patterns of haplotype within the block will be small, and the diversity of the block will be low.

Definition 1 (haplotype block diversity) *Given an interval [i, j] of a haplotype matrix A, a diversity function, $\delta : [i, j] \rightarrow \delta(i, j) \in R$ is an evaluation function measuring the diversity of the submatrix A(i, j).*

Diversity measurement usually reflects the activity of recombination events occurred during the evolutionary process. Generally, haplotype blocks with low diversity indicates conserved regions of genome.

Definition 2 (monotonic diversity) *A diversity function δ is said to be monotonic*
if, for any block (interval) I = [i, j] of A, it follows that $\delta(i , j) \leq \delta(i, j)$ whenever $[i , j] \subset [i, j]$; that is, the diversity of any subinterval of I is always no larger than the
diversity of I.

It is easily verified that many diversity functions, including the diversity function $\delta_D(S)$ defined by (2.2), are monotonic. However, the evaluative function of common haplotype proposed by Patil et al. [3] will not satisfy the monotonic property when the haplotype sample has missing data. For example, in Figure 5, it is a small portion of human Chromosome 21 haplotype sample provided by Patil, here *n* denotes the missing data. We can find that the coverage of common haplotype of interval [21900,21907] is 9/10, more than 80%. Therefore, according to the definition proposed by Patil et al., it is a feasible haplotype block. On the other hand, the coverage of common haplotype of interval [21902,21907] is 3/7, less than 80%, so it is not a feasible haplotype block. Note

that interval [21900,21907] and interval [21902,21907] are two intervals terminated at the same SNP locus, and interval [21900,21907] which has more SNPs is a feasible haplotype block but interval [21902,21907] is not.

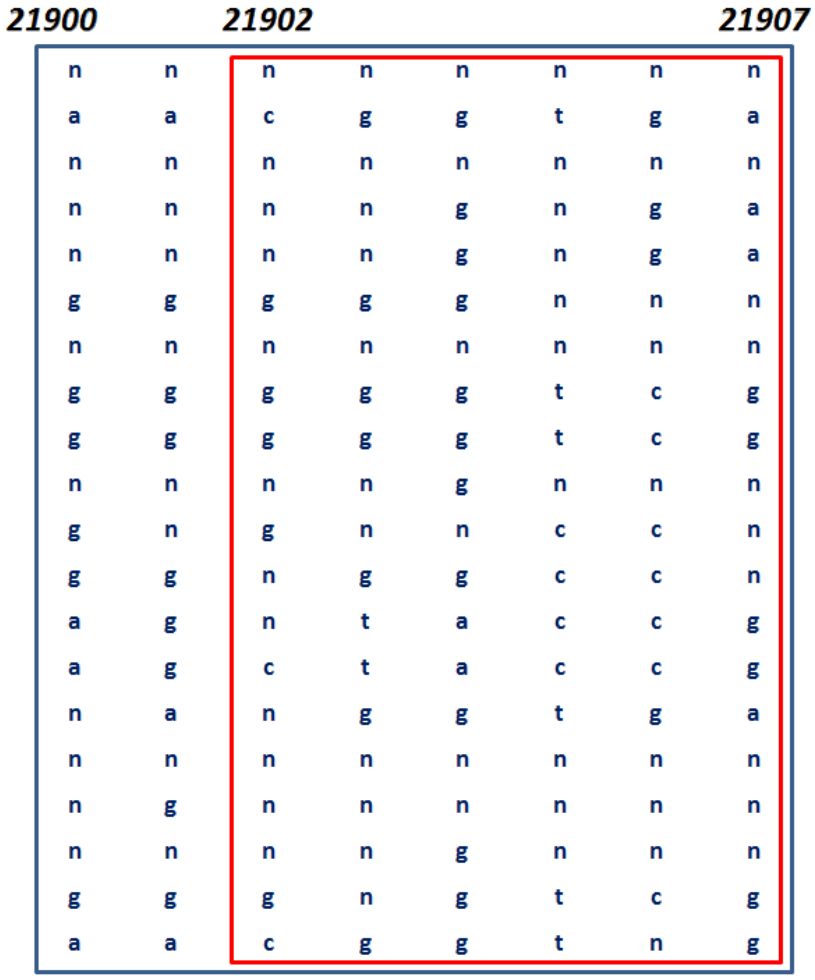

Figure 5: The evaluative function of common haplotype does not satisfy the mono property when the haplotype sample has missing data.

2.4 Hadoop MapReduce framework

Hadoop [11] is a software framework for coordinating computing nodes to process distributed data in parallel. Hadoop adopts the map/reduce parallel programming model, to develop parallel computing applications. The standard map/reduce mechanism has been applied in many successful Cloud computing

service providers, such as Yahoo, Amazon EC2, IBM, Google and so on. An application developed by Map/Reduce is composed of Map stage and Reduce stage (optionally). Figure 6 illustrates the Map/Reduce framework [12]. Input data will be split into smaller chunks corresponding to the number of Maps. Output of Map stage has the format of <*key, value*> pairs. Output from all Map nodes, <*key, value*> pairs, are classified by *key* before being distributed to Reduce stage. Reduce stage combines *value* by *key*. Output of Reduce stage are <*key, value*> pairs where each *key* is unique.

Hadoop cluster includes a single master and multiple slave nodes. The master node consists of a *jobtracker, tasktracker, namenode,* and *datanode.* A slave node, as computing node, consists of a datanode and tasktracker. Figure 7 illustrates the Hadoop cluster architecture. The jobtracker is the service within Hadoop that farms out Map/Reduce tasks to specific nodes in the cluster, ideally the nodes that have the data, or at least are in the same rack. A tasktracker is a node in the cluster that accepts tasks; Map, Reduce and Shuffle operations from a jobtracker.

Hadoop Distributed File System (HDFS) is the primary file system used by Hadoop framework. Each input file is split into data blocks that are distributed on datanodes. Hadoop also creates multiple replicas of data blocks and distributes them on datanodes throughout a cluster to enable reliable, extremely rapid computations. The namenode serves as both a directory namespace manager and a node metadata manager for the HDFS. There is a single namenode runnung in the HDFS architecture.

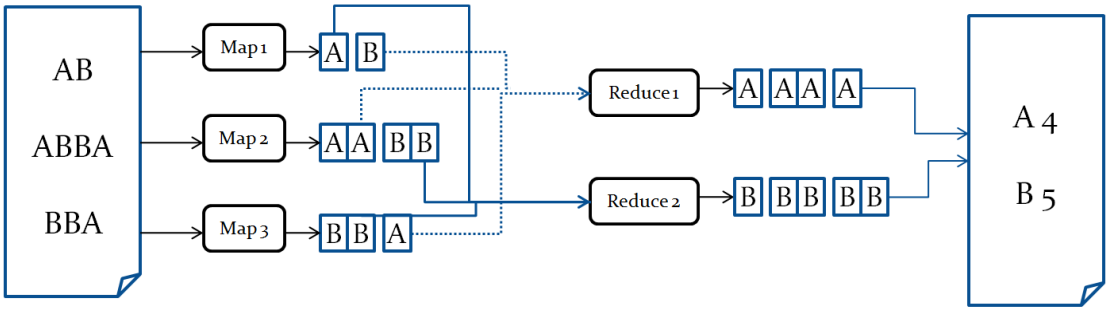

Figure 6: Example of counting words on MapReduce framework. The input data set {(A, B), (A, B ,B ,A),(B, B ,A)} are split into three maps, and each map process its data. Reduce process the data with same key. The output is number of word "A" and "B"

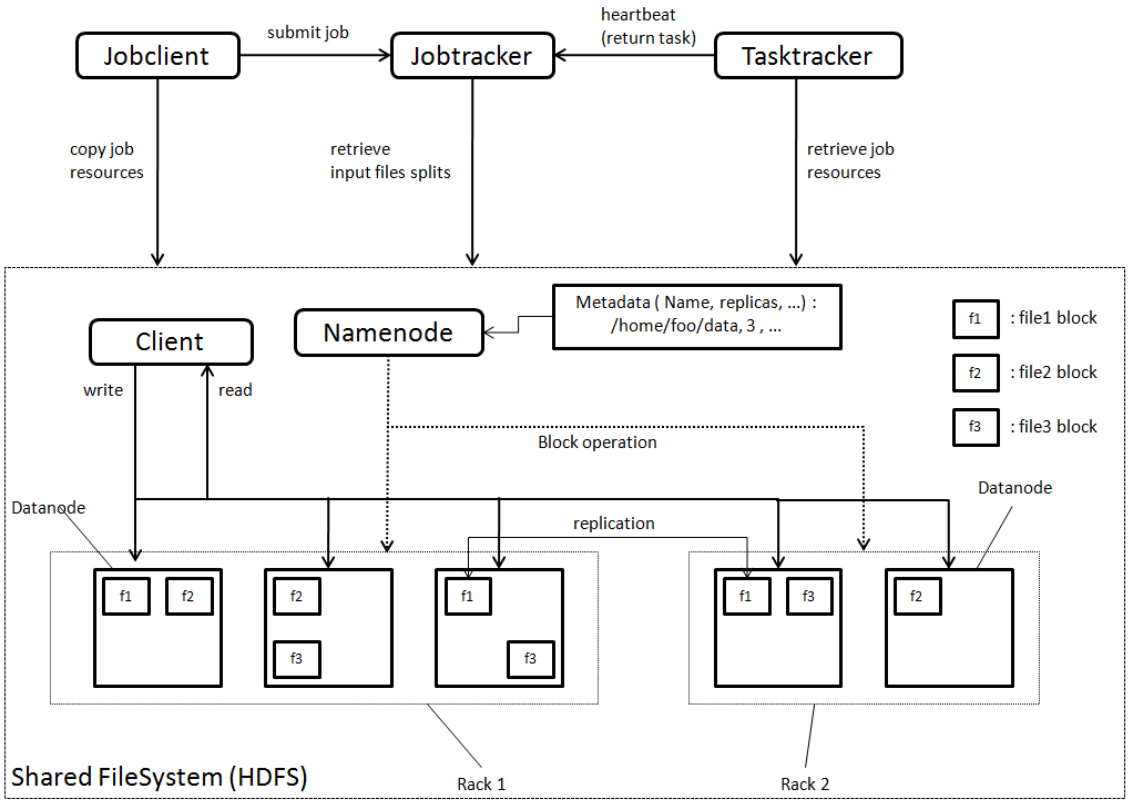

Figure 7: Hadoop Cluster architecture

Chapter 3

Algorithms for Longest Blocks Partitioning Using *k* Blocks of TagSNPs

In this chapter, we show a dynamic programming algorithms to partition longest blocks with constraint on diversity.[27][27] That is, we want to find the longest segmentation *S* consisted of *k* haplotype blocks with the diversity of each block is less than a diversity limit *D*. The problem definition is shown in Problem 1.

Problem 1 (*k*-longest-blocks) *Given a haplotype matrix A and a diversity upper limit D, find a segmentation consisted of k feasible blocks such that the diversity of each block is less than D and the total length is maximized. That is, output the set* $S = \{B_1, B_2, \ldots, B_k\}$, *with* $\delta(B) \leq D$ *for each* $B \in S$, *such that* $|B_1| + |B_2| + \bullet\bullet\bullet + |B_k|$ *is maximized.*

3.1 The Preprocessing

Before finding the longest *k* blocks, we first find the good partner of each SNP site to simplify the block selection. Because the diversity value of a candidate block must smaller than the diversity limit *D*, we can find the left farthest site (left good partner) $j = L[i]$ for each SNP marker *i* so that $[j, i]$ is the longest feasible block ended at site *i*. We can use techniques of suffix tree [28, 29] and lowest common ancestor (LCA) [30] to create a event list [31] that

represents the diversity variation of whole haplotype matrix. Assume that the diversity function we use is a monotonic non-decreasing function from [1..*n*; 1..*n*] to the unit real interval [0, 1]; that is, $0 \leq \delta(j, k) \leq \delta(j, k) \leq 1$ whenever $[j, k] \subset [j, k]$.

Therefore, if we know $j = L[i]$ is the left good partner of site *i*, the left good partner $L[i - 1]$ can be found at left side of $L[i]$; the idea is shown in Figure 8. All good partners for each SNP site can be found by scanning n diversity values. In the other case, if we use a non-monotonic diversity function such as the coverage of common haplotype proposed by Patil et al. [3], all good partners also can be calculated by scanning entire event list.

Note that the size of the event list is $m \times n$. The time complexity for creating the event list is $O(mn)$, so the preprocessing of the good partners of all SNP loci is $O(mn)$, linear proportional to the input size of haplotype matrix, time, regardless what kinds of diversity function we used.

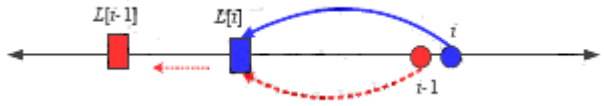

Figure 8: The calculation of the left good partners.

3.2 Dynamic Programming Algorithms

Given a haplotype matrix *A* and a diversity upper limit *D*, let $S = \{B_1, B_2, \ldots, B_k\}$ be a segmentation of *A* with $\delta(B) \leq D$ for each $B \in S$. The length of *S* is the total length of all blocks in *S*; i.e., $(S) = |B_1| + |B_2| + \bullet \bullet \bullet + |B_k|$. Our objective is to find a segmentation consisted of *k* feasible haplotype blocks such that the total length (*S*) is maximized. Given *A* and *D*, first we consider the most general

form of the problem and define the block length evaluation function.

f (k, i, j) = max{ (S) | S a feasible segmentation of A(i, j) with k blocks}

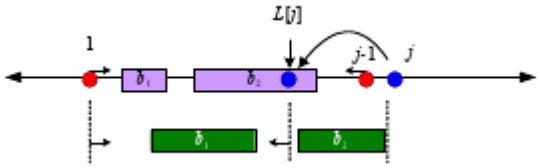

Figure 9: The idea of the recurrence relation used to compute $f(k, 1, n)$

Note that the *k*-longest-blocks problem asks to find the value $f(k, 1, n)$. After the left farthest sites, $L[j]$'s, are calculated, the answer can be found in $O(nk)$ time. The idea behind the dynamic programming formula is illustrated at Figure 9.

It can be verified that

$$f(k, 1, j) = \max\{f(k, 1, j-1), f(k-1, 1, L[j]-1) + j - L[j] + 1\}$$

That is, the *k*-th block of the maximal segmentation *S* in $[1, j]$ either does not include site *j*; otherwise, the block $[L[j], j]$ must be the last block of *S*. Note that $f(k, 1, j)$ can be determined in $O(1)$ time when $f(k-1, 1, \bullet)$'s and $f(k, 1, 1..(j-1))$'s are ready. It follows that f $(k, 1, \bullet)$'s can be calculated from $f(k-1, 1, \bullet)$'s, totally in $O(n)$ time. Thus a computation ordering from $f(1, 1, \bullet)$'s, $f(2, 1, \bullet)$'s, . . ., to $f(k, 1, \bullet)$'s leads to the result of $O(nk)$ time complexity.

Theorem 1 (*k*-longest-blocks) *Given a haplotype matrix A and a diversity upper limit D, find a segmentation consisted of k feasible blocks such that the total length is maximized can be done in O(nk) time after a linear time preprocessing.*

We show the dynamic programming algorithm for haplotype blocking in Figure 10. Note that we assume the diversity function used here is monotonic.

FINDBLOCK(k, i, j) ▷ Find a segmentation consisted of k feasible blocks in $[i, j]$

Input: Interval $[i, j]$ and number of blocks k.

▷*Left farthest site* for each marker site must be prepared.

Output: The length of the longest segmentation consisted of k feasible blocks in $[i, j]$.

1 **for** $y \leftarrow i$ **to** j **do** ▷ Initiate the boundary condition of $f(k, i, j)$.

2 $length[0, y] \leftarrow 0$

3 **for** $x \leftarrow 1$ **to** k **do** ▷ Initiate the boundary condition of $f(k, i, j)$.

4 $length[x, i] \leftarrow 1$

5 **for** $x \leftarrow 1$ **to** k **do**

6 **for** $y \leftarrow i$ **to** j **do**

7 **if** $L[y] \leq i$ **then** ▷ $L[y] \notin [i, j]$, exceeding the boundary region.

8 $length[x, y] \leftarrow y - i + 1$

9 **else**

10 $temp1 \leftarrow length[x, y - 1]$

11 $temp2 \leftarrow length[x - 1, L[y] - 1] + y - L[y] + 1$

12 $length[x, y] \leftarrow \max\{temp1, temp2\}$

13 **return** $length[k, j]$

Figure 10: The $O(nk)$ time algorithm for the longest haplotype blocks partitioning.

3.3 TagSNPs Selection

For each block, we want to minimize the number of SNPs that uniquely distinguish at least 80% (the α parameter) of the unambiguous haplotypes in the block. Those SNPs can be interpreted as a signature of the haplotype block partition. They are referred to as tagSNPs that are able to capture most of the

haplotype diversity, and therefore, could potentially capture most of the information for association between a trait and the marker loci [32].

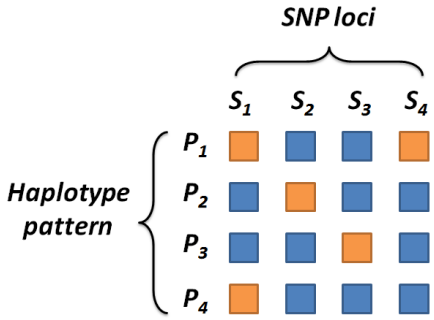

Figure 11: An example of a longer block but required less tagSNPs.

Our strategy for selecting the tagSNPs in haplotype blocks is as the following. First, the common haplotypes are grouped into *k* distinct patterns in each block. After the missing data are assigned, as explained in the next subsection, we decide the least number of groups needed such that haplotypes in these groups contain at least 80% (α) of the unambiguous haplotypes in the block. Finallly, we select a loci set which consists of the minimum number of SNPs on the haplotypes such that each pattern can be uniquely distinguish. Exhaustive searching methods are used very efficiently here since the number of tagSNPs needed for each block is usually modest in the situation. The exhaustive searching algorithm enumerates next γ-combination in lexicographic order to generate the next candidate tagSNP loci set until each pattern can be uniquely distinguish.

Chapter 4

Using MapReduce framework to compute block diversity in parallel

In the previous chapters, we can find out the time complexity of a dynamic programming algorithm to partition longest blocks with *k* blocks is *O*(*nk*), but prepare the diversity scores, the time spent in the calculation of diversity scores is greater than the dynamic programming algorithm with selecting longest blocks.

In this chapter, we show a method to parallel computing diversity scores of all interval in a $m \times n$ matrix. According MapReduce framework model, we want to divide the diversity computation into several maps.[33]

4.1 The Preprocessing

Patil et al. defined a haplotype block as a region where at least 80% of observed haplotypes within a block must be common haplotype. The diversity defined by

$$\delta s(A) = 1 - \frac{C}{U} = \frac{S}{U}$$

Here U denotes the number of unambiguous haplotype, C denotes the number of common haplotypes, and S denotes the number of singleton haplotypes.

To find the good partner of each SNP site, we must find out the diversity of

block *A* just larger than a boundary *D*, for Patil. defined, $\delta_S(A) \leqq 20\%$. This idea is illustrated in Figure 12.

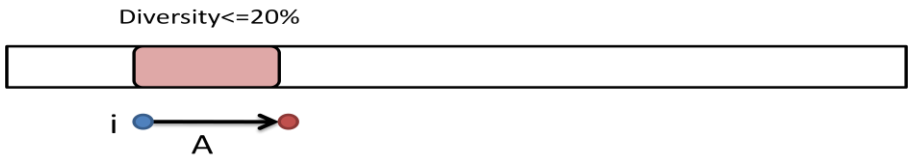

Figure 12: Find good partner by $\delta_S(A) \leqq 20\%$

Without computing all interval diversity, Patil. proposed that the haplotype block can be found within 300bp and 500bp. Therefore, we can find the good partner of each SNP site by increase the block size to 300bp or 500bp, the idea is shown in Figure 13.

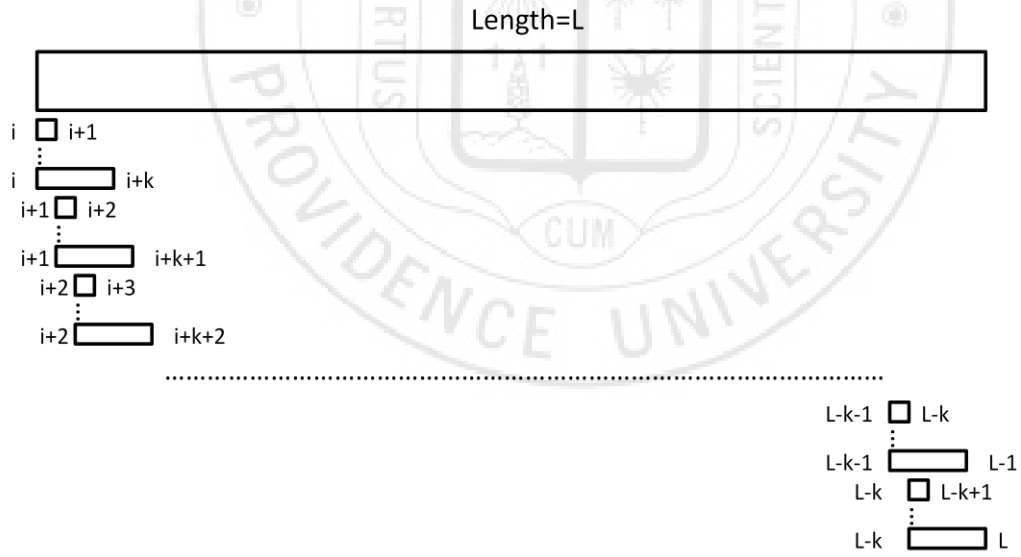

Figure 13: Find all the block diversity within k bp

4.2 Divide work

Figure 14 illustrates a haplotype matrix which divided into three parts, the

length of each part is $L/3$, and haplotype block can be found within k bp. Assume that the first part of block $\delta(1, L/3)$ start calculated all interval diversity, we find that the diversity computation processing to blocks $\{\delta(L/3\text{-k+1}, L/3\text{-k+2}), ..., \delta(L/3\text{-k+1}, L/3\text{+1})\}$, the information of spot $(L/3+1)$ is needed. It is easily verified that the extra information of block $\delta(L/3+1, L/3+k)$ is needed when calculating all interval diversity of block $\delta(1, L/3)$.

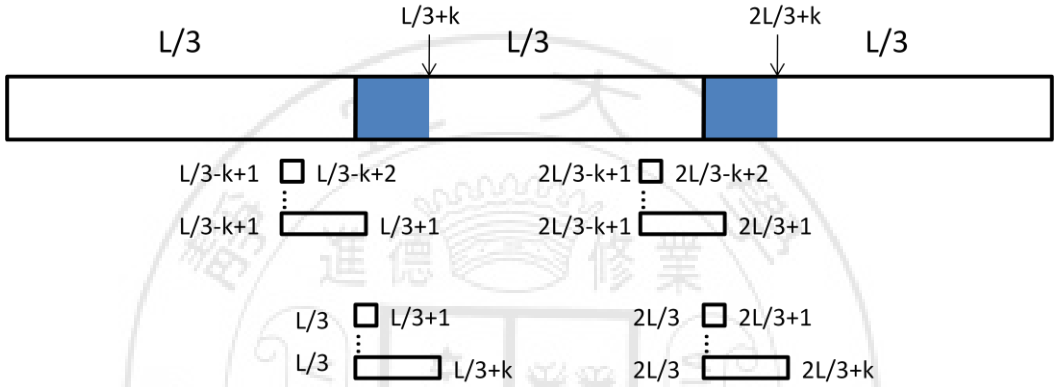

Figure 14: The needed blue area information for calculating k bp diversity

We can proposed that calculating all interval diversity of a haplotype block $\delta(1,L/3)$ only need two neighbor block, $\{\delta(1, L/3), \delta(L/3+1, L/3+k)\}$, and therefore a map can calculate diversity of $\delta(1,L/3)$ when it received information of $\delta(1, L/3+k)$. The idea is illustrated at Figure 15.

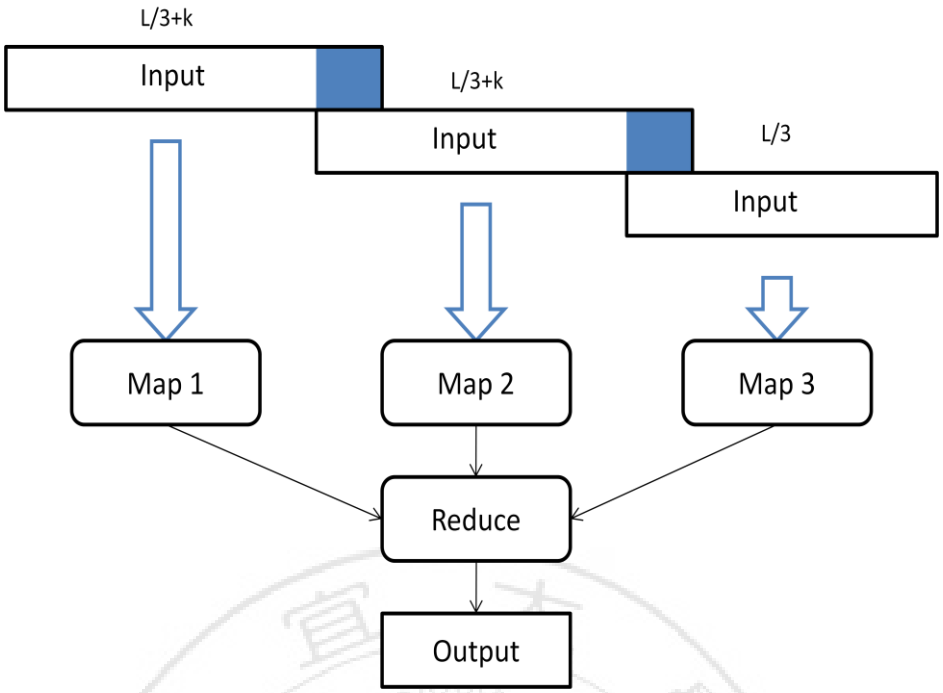

Figure 15: The idea of dividing question into independent parts for maps

4.3 Parallel work

Figure 16 illustrates the MapReduce framework for the block partitioning and selection scheme. Assume that the number of map operation is N and the pattern length is L, the input $N \times L$ haplotype matrix is split into L/N chunks. Each map calsulates the diversity scores of each block within the chunk where the map operation is responsible. Thus the output <*key*, *value*> pairs for each Map are <(*block start number, block end number*), diversity score> pairs.

The map_i calculates diversity scores of blacks $\{\delta(i \cdot N/L, i \cdot N/L), \delta(i \cdot N/L, i \cdot N/L+1), ..., \delta(i \cdot N/L+ N/L, i \cdot N/L+ N/L)\}$.

Therefore, each map has $(N/L)^2$ diversity scores. Reduce stage performs haplotype block selection algorithm. In our algorithm, just one reduce operation is needed in the reduce stage. Since the selection is a linear time algorithm, it is

not necessary to perform the computation in parallel. The reduce operation finds the longest block by merging blocks with the interesting diversity scores.

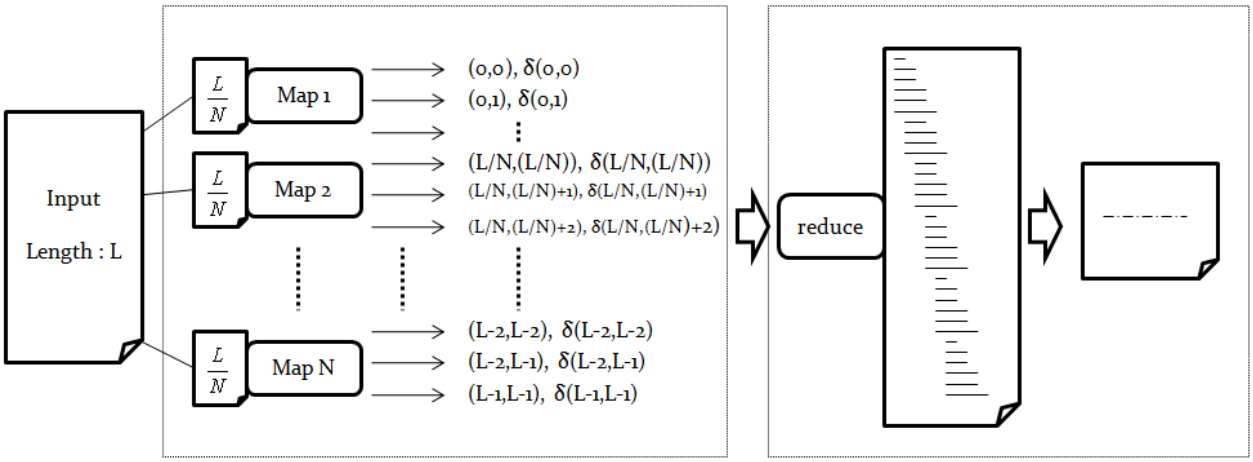

Figure 16: Haplotype block partitioning and selection on MapReduce framwork.

Chapter 5

Experiments

5.1 Experimental environment and data source

All of the experiments were performed on two IBM blade servers within our Cloud Computation Laboratory. Each server is equipped with two Quad-Core Intel Xeon 2.26GHz CPU, 24G RAM, and 296G hard disk running under the operation system Ubuntu version 10.4 with Hadoop version 0.2 MapReduce platform. Under the current system environment, we control the server execution processes by up to 8 map operations and 8 reduce operations and the total number of the map/reduce operations are up to 16 respectively.

The SNP haplotype data sources are gathered from the International HapMap Project (http://hapmap.ncbi.nlm.nih.gov/), which is a multi-country effort in order to identify and catalog genetic similarities and differences in human beings. This project collects many genetic sequences of different individuals. We downloaded the sequence data (Chromosome 1) from the HapMap3 Genome Browser release #2-African ancestry in Southwest USA (ASW). ASW includes 136 Chromosome 1 (chr 1) sequences (patterns) and the length of SNP is 116,416. These sequences are treated as the input data for our experiments. In the experiments, we applied common diversity to calculate diversity scores of blocks.

5.2 Implement

Figure 17 illustrate the flowchart of the parallel work. At first, the input file is spilt and stored into Hadoop HDFS through the Namenode from the local client. And then, the job is send by streaming API, several Maps are assigning to process diversity computing in parallel. Finally, the results of diversity will be collected by Reduce, and storing on HDFS.

When we got those results of diversity on HDFS, we can found blocks and tags of Haplotypes by the Longest Blocks Partitioning Using k Blocks of TagSNPs algorithm. At this step, we wouldn't spend a lot time.

The overall parallel design is how to divide haplotype file into few arguments and few file segments, each map only receive those data and could processing in independent. The main code of divide will present in appendix A.

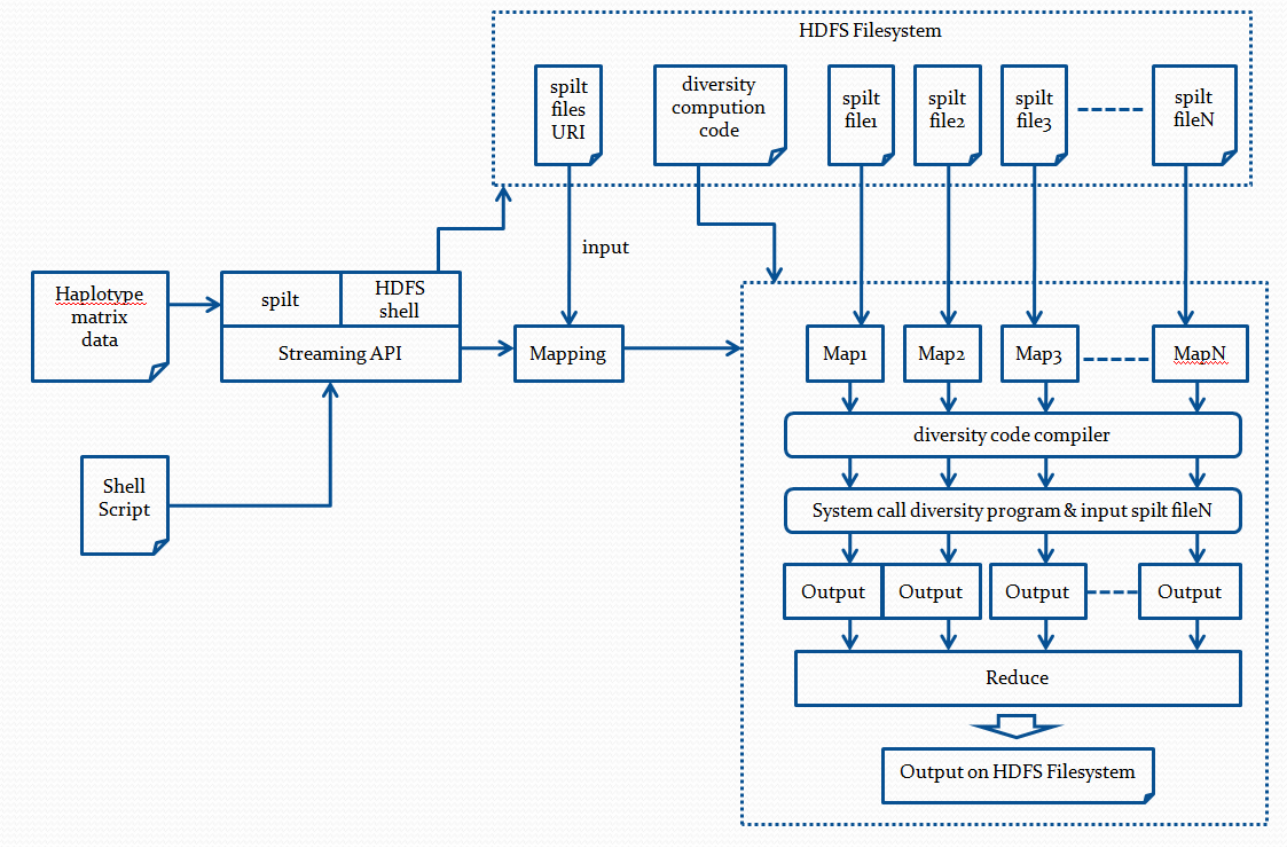

Figure 17:Hadoop Programming Flowchart.

5.3 Experimental results

To assess the performance of the proposed Hadoop MapReduce algorithm, we compared the computational time between various sequence data and various number of map/reduce operations. The sequential algorithm [32] has been proved as an efficient algorithm than other algorithms. Two factors, number of patterns and the length of patterns, affect the performance of sequential algorithm and the haplotype block can be found within 300bp and 500bp. Therefore, the block size can be 300bp and 500bp.

The diversity scores are calculated according to their corresponding block sizes; these scores are {$\delta(1, 1)$, $\delta(1, 2)$, ..., $\delta(1, 500)$, $\delta(2, 2)$, ..., $\delta(2, 501)$, $\delta(3, 3)$, ..., $\delta(L, L)$}. In Figure 18 and Figure 19, the block sizes are 300bp and 500bp, respectively.

The experimental results reveal that the computational time is effectively reduced when more map operations are deployed. Two and four map operations almost improve the computation time by factors of two and four times accordingly, comparing to the original sequential algorithm, respectively. Moderate enhancements between 8 and 16 map operations are observed in all experiments, since the size of data set split by 8 is similar to that by 16. Figure 20 illustrates the computation efficiency can be effectively improved proportional to the number of processors being used.

In Figure 18 and Figure 19, it is observed that computational time increases corresponding to number of pattern and sequence length. The computational time for our algorithm with block size 300bp is less than that with block size 500bp. More patterns and longer sequence length lead to higher computational cost. These experimental results are corresponding to the algorithm analysis in

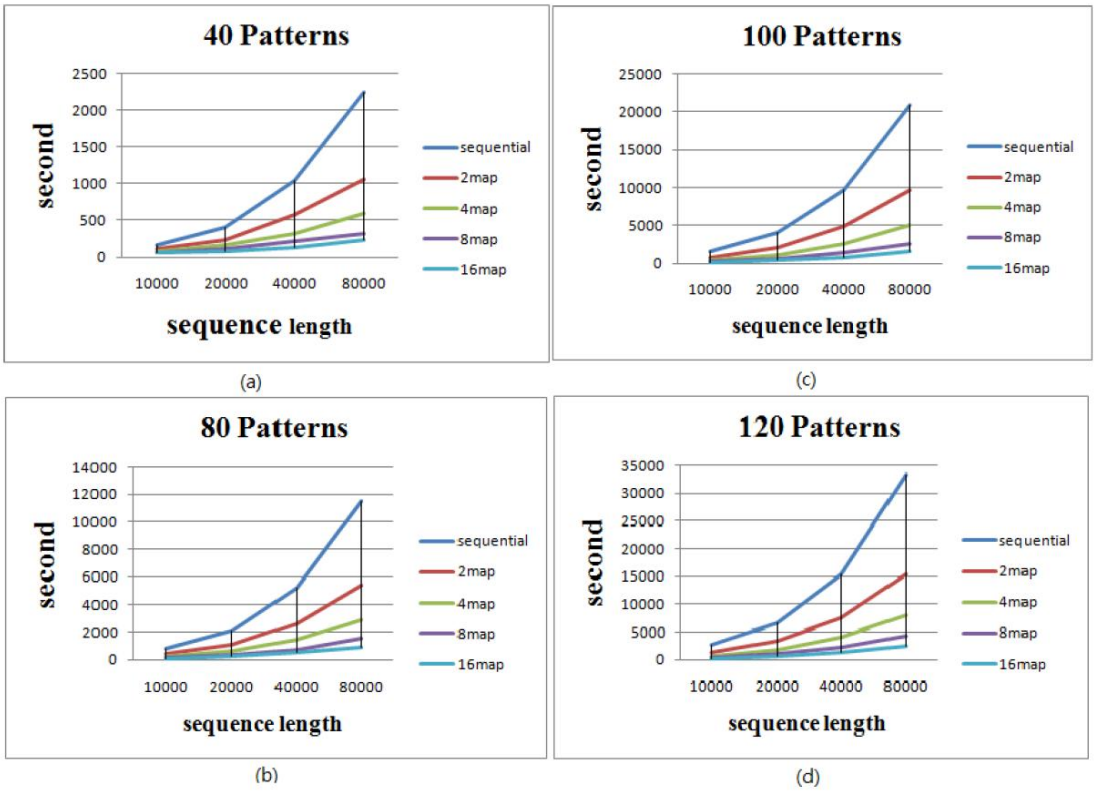

Figure 18: Performance comparsion between sequential haplotype block selection and MapReduce haplotype block selection with block size 300bp.

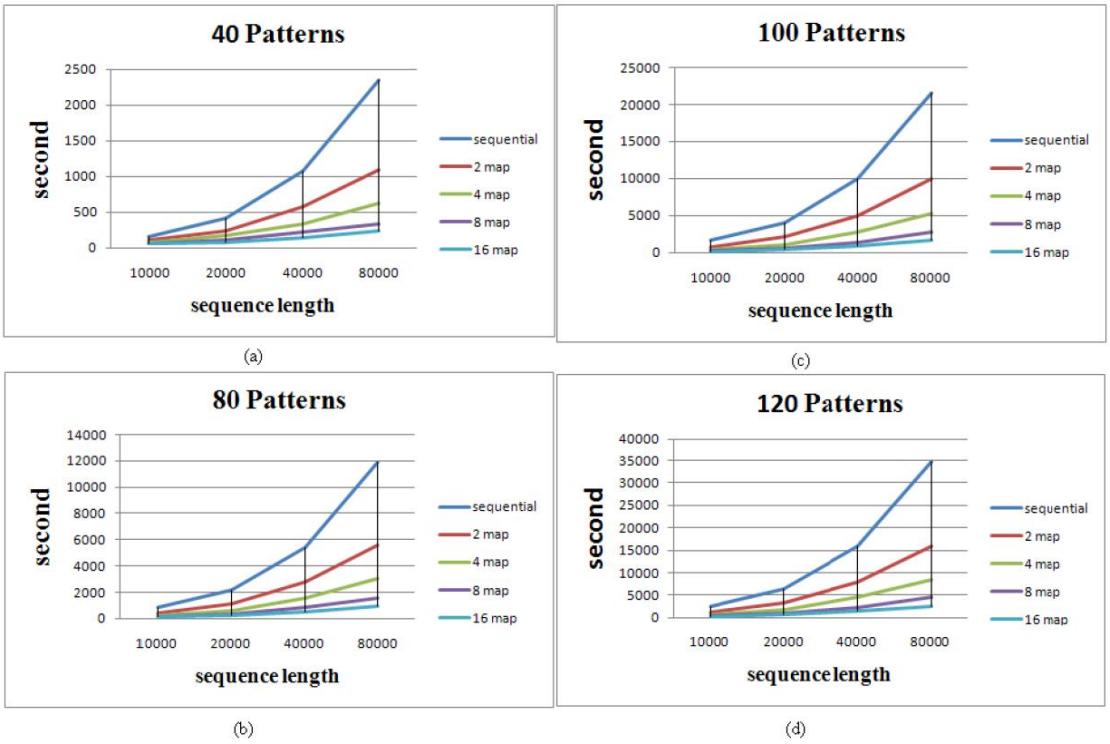

Figure 19: Performance comparsion between sequential haplotype block selection and MapReduce haplotype block selection with block size 500bp.

previous section.

Finally, we have the data that thought parallel diversity computing by Hadoop MapReduce, and we can using the Longest Blocks Partitioning Using k Blocks algorithm to analysis it.

Figure 21 illustrate a block count result for the whole length of ASW_Chr1, and it is covered by 792 blocks which diversity ≤ 0.8. We take a statistics in block length which interval equals 10, and count the block number of each interval.

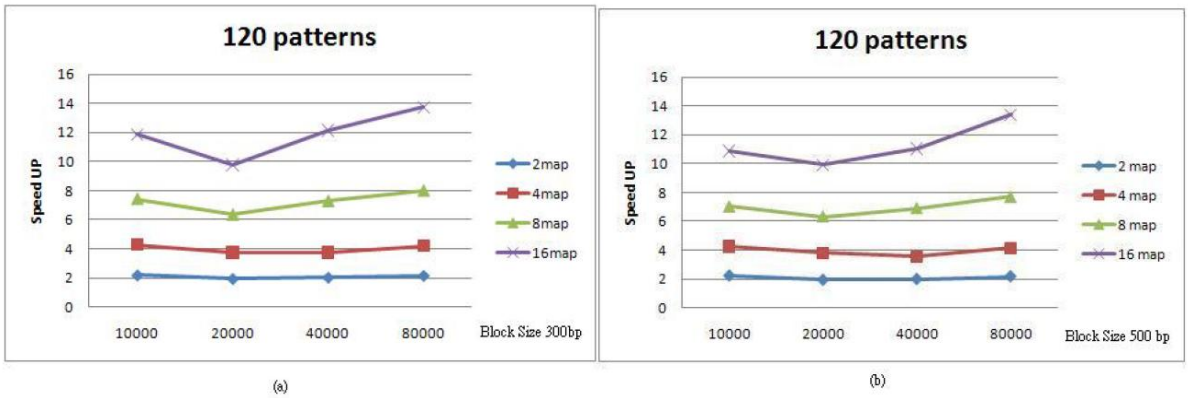

Figure 20: Speed up comparisons for MapReduce haplotype block selection over sequential haplotype block selection. (a) illustrates the speed up with block size 300bp. (b) illustrates the speed up with block size with 500bp

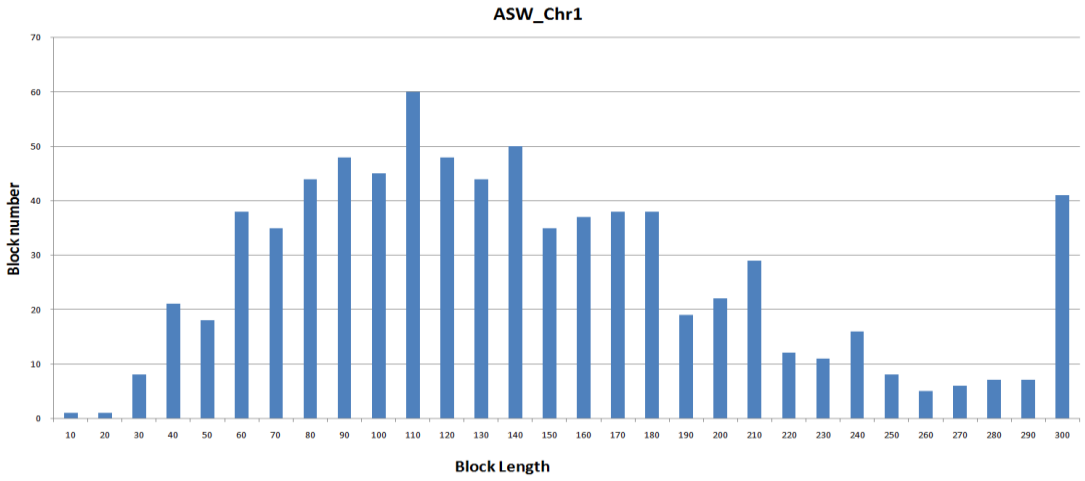

Figure 21: the block length statistics of ASW_Chr1

Chapter 6

Conclusion

6.1 Conclusion

Studying on SNP and haplotype blocks can assist biomedical researchers to detect inherited diseases and contribute to classify the race of human and researching on species evolution. The traditional methods for detecting haplotype blocks are based on dynamic programming approach. With the abundance of bioinformatics data that are all too common these days, the time-consuming traditional sequential methods require imminent assistance of the emerging parallel processing methodology.

Here in this paper we discuss how we develop the parallelized frame works improving our original dynamic programming algorithms, based on Hadoop map/reduce framework. The haplotype block partitioning copes with the problem of finding the minimum nubmber of representative SNPs required to account for most of the haplotype block quality in each block. Due to the fault tolerance of Hadoop, the jobs are just re-submitted to other nodes if the node is failure. This property is useful for analyzing large amount of sequence data since the job will not be stopped by node's fail. The experimental results show that the proposed algorithm can decrease the computational cost significantly.

6.2 Future work

In this thesis, we compared the performance between various sequence lenghts and pattern numbers. We also compared the performance between different block sizes. In the future, we will apply more diversity functions to the parallel algorithms to provide more perspectives for biologists to analyze these SNP data and investigate the relations of haplotype block selection between various block size and diversity measurements.

Reference

[1] International HapMap Project. http://www.hapmap.org/index.html.en.

[2] J. Craig Venter, Mark D. Adams, Eugene W. Myers, et al. The Sequence of the Human Genome. Science, 291(5507):1304–1351, 2001.

[3] N. Patil, A. J. Berno, D. A. Hinds, et al. Blocks of limited haplotype diversity revealed by high resolution scanning of human chromosome 21. Science, 294:1719–1723, 2001.

[4] S. B. Gabriel, S. F. Schaffner, H. Nguyen, et al. The structure of haplotype blocks in the human genome. Science, 296(5576):2225–2229, 2002.

[5] M. J. Daly, J. D. Rioux, S. F. Scha?ner, T. J. Hudson, and E. S. Lander. High-resolution haplotype structure in the human genome. Nature Genetics, 29:229–232, 2001.

[6] M. Olivier, V. I. Bustos, M. R. Levy, et al. Complex High-Resolution Linkage Disequilibrium and Haplotype Patterns of Single-Nucleotide Polymorphisms in 2.5 Mb of Sequence on Human Chromosome 21. Genomics, 78, Nov.

[7] Russell Schwartz, Andrew G. Clark, and Sorin Istrail. Methods for Inferring Block-Wise Ancestral History from Haploid Sequences. In WABI, pages 44–59, 2002.

[8] G. C. Johnson, L. Esposito, B. J. Barratt, et al. Haplotype tagging for the identification of common disease genes. Nat Genet., 29(2):233 – 7, Oct

2001.

[9] K. Zhang, M. Deng, T. Chen, M.S. Waterman, and F. Sun. A dynamic programming algorithm for haplotype block partitioning. In The National Academy of Sciences, volume 99, pages 7335–7339, 2002.

[10] J. D. Rioux, M. J. Daly, M. S. Silverberg, K. Lindblad, H. Steinhart, Z. Cohen, et al. Genetic variation in the 5q31 cytokine gene cluster confers susceptibility to Crohn disease. Nature Genetics, 29:223–228, 2001.

[11] Hadoop - Apache Software foundation project home page [http://hadoop.apache.org/]

[12] Taylor, R.C., 2010, An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. BMC Bioinformatics, 11:S1.

[13] Dean, J., Ghemawat, S., 2010, MapReduce: A Flexible Data Processing Tool. Communications of the ACM, 53, 72-77.

[14] Schatz, M., 2009, Cloudburst: Cloudburst: highly sensitive read mapping with MapReduce. Bioinformatics, 25, 1363-1369.

[15] D. Clayton. Choosing a set of haplotype tagging SNPs from a larger set of diallelic loci. Nature Genetics, 29(2), 2001.

[16] K. Zhang, Z.S. Qin, J.S. Liu, T. Chen T, M.S. Waterman, and F. Sun. Haplotype

[17] block partitioning and tag SNP selection using genotype data and their applications to association studies. Genome Res., 14(5):908–916, 2004.

[18] J.D. Wall and J.K Pritchard. Haplotype blocks and linkage disequilibrium in the human genome. Nature Reviews Genetics, 4(8):587–597, 2003.

[19] R. R. Hudson and N. L. Kaplan. Statistical properties of the number of recombination events in the history of a sample of DNA sequences. Genetics, 111:147–164, 1985.

[20] N. Wang, J.M. Akey, K. Zhang, R. Chakraborty, and L. Jin. Distribution of recombination crossovers and the origin of haplotype blocks: the interplay of population history, recombination, and mutation. Am. J. Human Genetics, 71:1227–1234, 2002.

[21] M. Nordborg and S. Tavare. Linkage disequilibrium: what history has to tell us. Trends in Genetics, 18, Feb.

[22] R. C. Lewontin. THE INTERACTION OF SELECTION AND LINKAGE. I. GENERAL CONSIDERATIONS; HETEROTIC MODELS. Genetics, 49(1):49–67, 1964.

[23] Eric C. Anderson and John Novembre. Finding Haplotype Block Boundaries by Using the Minimum-Description-Length Principle. Am. J. of Human Genetics, 73:336–354, 2003.

[24] M. Koivisto, M. Perola, R. Varilo, W. Hennah, J. Ekelund, M. Lukk, L. Peltonen, E. Ukkonen, and H. Mannila. An MDL method for ?nding haplotype blocks and for estimating the strength of haplotype block boundaries. In 8th Pacific Symposium on Biocomputing (PSB), pages 502–513, 2003.

[25] G. Greenspan and D. Geiger. Model-based inference of haplotype block

variation. In Seventh Annual International Conference on Computational Molecular Biology (RECOMB), 2003.

[26] W.H. Li and D. Graur. Fundamentals of Molecular Evolution. Sinauer Associates, Inc, 1991.

[27] Yaw-Ling Lin, Guan-Jie Hua, Wen-Pei Chen. Block Partition and Tag Selection in Human SNP Haplotype. Computer Society of the Republic of China, 21(3), 59-69, 2010.

[28] D. Gusfield. Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. Cambridge University Press, 1997.

[29] Esko Ukkonen. On-Line Construction of Su?x Trees. Algorithmica, 14(3):249–260, 1995.

[30] D. Harel and R. E. Tarjan. Fast algorithms for ?nding nearest common ancestors. SIAM Journal on Computing, 13(2):338–355, 1984.

[31] Wei-Shun Su. A Study on SNP Haplotype Blocks. Master's thesis, Providence University, Jun 2006.

[32] Chapman, J.M., Cooper, J.D., Todd, J.A., Clayton, D.G., 2003, Detecting disease associations due to linkage disequilibrium using haplotype tags: a class of tests and the determinants of statistical power. Hum. Hered., 56, 18-31.

[33] C. L. Hung, Y. L. Lin, G. J. Hua, Y. C. Hu, "CloudTSS: A TagSNP Selection Approach on Cloud Computing," Grid and Distributed Computing, Communications in Computer and Information Science, (T. H. Kim et. Al., Eds.), Springer-Verlag, Vol. 261, pp. 525-534, 2011. [EI]

Appendix A

divpart.pl (divide file and call streaming API to parallel)

```
001  #$ARGV[0] = "ASW_Chr21"; #file name
002  #$ARGV[1] = 4;                #divide file number
003  #$ARGV[2] = 0.8;              #diversity
004  #$ARGV[3] = 500;              #latest good partner
005  $HH = "/opt/hadoop";          #Hadoop home dir;
006
007  if (!@ARGV){
008          print "Usage : divpart [file] [part] [diversity] [threshold]\n";
009          exit(2);
010  }
011
012  $file = $ARGV[0];
013  $cut = $ARGV[1];
014  $div = $ARGV[2];
015  $bp = $ARGV[3];
016
017  #test input file exist
018  if(system("test -e $file")){
019          print "Error: Input file not exist\n";
020          exit(2);
021  }
022  #find input file matrix size
023  open(IN,"$file");
024  $firstline = <IN>;
025  chomp($firstline);
026  $n = length($firstline);
027  $m = 1;
028  while(<IN>){
029          $m ++;
030  }
031  close(IN);
032
```

```
033 $job='job_'.time();
034 $dir="/Div/$job";
035 mkdir $job;
036
037 #test the output file whether exist
038 print "----------Creat directory on Hadoop----------\n";
039
040 system("$HH/bin/hadoop fs -mkdir /Div");
041
042 $ret=system("$HH/bin/hadoop fs -test -e $dir");
043 if(!$ret){
044         print "Output file already exist, remove this file?(yes/no) : ";
045         while(1){
046                 $ret = <STDIN>;
047                 chomp($ret);
048                 if($ret =~ 'yes'){
049                         system("$HH/bin/hadoop fs -rmr $dir");
050                         last;
051                 }elsif($ret =~ 'no'){exit(0);
052                 }else{print "Type (yes/no) : "}
053 }}
054 $ret = system("$HH/bin/hadoop fs -mkdir $dir");
055 if(!$ret){
056     print "Create $dir on Hadoop\n";
057 }
058
059 open(INFO,">$job/file.info");
060 print INFO "file name : $file\n",
061             "file partition : $cut\n",
062             "pattem length(n) : $n\n",
063             "patten number(m) : $m\n",
064             "diversity : $div\n",
065             "threshold : $bp\n",
066             "\n";
067 close(INFO);
068 $ret = system("$HH/bin/hadoop fs -put $job/file.info $dir");
069 if(!$ret){
070                 print "Put file file.info into HDFS:$dir/file.info\n";
071         }
```

```
072
073 print "------------------------Partition file start-----------------------\n";
074
075 #determine how long per part
076 if($n%$cut==0){
077         $divlength = $n/$cut,"\n";
078 }else{
079         $divlength = int($n/$cut)+1,"\n";
080 }
081
082 #divide file and up into HDFS
083 open(CON,">$job/part_path");
084 $startspot = $n;
085 for($i = $cut ; $i > 0 ; $i--){
086         $length = $divlength + $bp;
087         if($startspot - $divlength - $bp < 0 ){
088                 $length = $startspot;
089                 $divlength = $length;
090         }
091        print "Partition block ",$startspot-$length,"-",$startspot," on file $part$i\n";
092         open(OUT,">$job/part$i");
093         print OUT $m," ",$length," ",$div," ",$startspot," ",$bp," ",$divlength,"\n";
094         open(IN,"$file");
095         while(<IN>){
096         chomp($_);
097                 print OUT substr($_,$startspot-$length,$length),"\n";
098         }
099         close(IN);
100         print OUT "\n";
101         close(OUT);
102         $startspot -= $divlength;
103
104         $ret = system("$HH/bin/hadoop fs -put $job/part$i $dir");
105         if(!$ret){
106                 print "Put file part$i into HDFS:$dir/part$i\n";
107         }
108         print CON "$dir/part$i\n";
109 }
110 close(CON);
```

```
111 print "\n";
112
113 #create metafile of divide file's path
114 print "----------------------General metafile----------------------------\n";
115 print "hadoop fs -put $job/part_path $dir\n";
116 $ret = system("$HH/bin/hadoop fs -put $job/part_path $dir");
117 if(!$ret){
118         print "Put file part_path into HDFS:$dir/part_path\n";
119 }
120 print "\n";
121
122 #put diversity compute program
123 print "----------------------Put divall.c into hadoop---------------------\n";
124 $ret = system("$HH/bin/hadoop fs -put divall.c $dir");
125 print "Put divall.c into HDFS:$dir/divall.c\n";
126
127 print "----------------------Timing start---------------------------------\n";
128 $t1 = time();
129 print "Start Time: ",scalar gmtime $t1;
130
131 print "\n";
132
133 #now Hadoop streaming running
134 print "----------------------Execution hadoop Streaming--------------------\n";
135 print "hadoop jar $HH/contrib/streaming/hadoop-streaming-*.jar \\
136 -D mapred.max.tracker.failures=100 \\
137 -D mapred.map.tasks.speculative.execution=false \\
138 -D mapred.reduce.task=0 \\
139 -D mapred.task.timeout=12000000 \\
140 -inputformat org.apache.hadoop.mapred.lib.NLineInputFormat \\
141 -output $dir/out \\
142 -mapper run.sh \\
143 -file run.sh \\
144 -input $dir/part_path\n";
145 print "\n";
146 system("$HH/bin/hadoop jar $HH/contrib/streaming/hadoop-streaming-*.jar -D
    mapred.max.tracker.failures=100 -D mapred.map.tasks.speculative.execution=false -D
    mapred.reduce.task=0 -D mapred.task.timeout=12000000 -inputformat
    org.apache.hadoop.mapred.lib.NLineInputFormat -output $dir/out -mapper run.sh -file run.sh -input
```

```
$dir/part_path");

$t2 = time();
print "Finish Time : ",scalar gmtime $t2;
$t3 = $t2-$t1;
print "\n";

print "----------Execution time----------\n";
print "Hadoop execution cost $t3 sec.\n";
open(OUT,">$job/exec_time");
print OUT "file : $file\n","size(wide*high) : $n * $m\n","cut : $cut\n","diversity : $div\n","threshold :
 $bp\n","hadoop execution time : $t3\n";
close(OUT);
system("$HH/bin/hadoop fs -put $job/exec_time $dir");

print "\n";
```

---

run.sh (each map processing this script)

```
#!/usr/bin/env bash
read offset file

#random a number
declare -i ran=$(echo $RANDOM$RANDOM)

#hadoop home path
HH=/opt/hadoop

#filter path
dir=`echo $file | cut -d '/' -f 1,2,3`
prog=$dir/divall.c
part=`echo $file | cut -d '/' -f 4`

#download prog & compiler
$HH/bin/hadoop fs -cat $prog > /tmp/divall$ran.c
g++ /tmp/divall$ran.c -o /tmp/divall$ran
```

```
018
019 #exec
020 $HH/bin/hadoop fs -cat $file | /tmp/divall$ran > /tmp/$ran-ok
021
022 #output in HDFS
023 $HH/bin/hadoop fs -put /tmp/$ran-ok $dir/out/$part
024
025 #clean
026 rm /tmp/$ran-ok
027 rm /tmp/divall$ran
028 rm /tmp/divall$ran.c
```
