논문 19-44-05-28 The Journal of Korean Institute of Communications and Information Sciences '19-05 Vol.44 No.05
https://doi.org/10.7840/kics.2019.44.5.1024

MuTect 기반 체세포 변이 위치 탐색 기술 최적화

나 병 국◆, 윤 성 로°

Optimization for Detection of Somatic Mutations Based on MuTect

Byunggook Na◆, Sungroh Yoon°

요 약

다양한 질병의 원인, 특히 암의 원인을 유전체 데이터로부터 찾아내는 체세포 변이 위치 탐색 기술은 유전체 데이터 분석 분야에서 중요한 역할을 한다. 본 논문에서는 대중적으로 쓰이는 체세포 변이 위치 탐색 도구 중 MuTect를 검토하고, MuTect의 느린 수행 속도의 원인에 대해 분석하였다. 이는 MuTect 알고리즘이 Pileup 오퍼레이터를 중복 호출하며, 호출된 정보에 대한 가공 역시 중복 수행되기 때문이다. 따라서, 관찰된 MuTect의 비효율성을 해결하기 위해, 메모리 상에 로드되어 있는 유전체 데이터의 데이터 구조를 바꾸는 데이터 재배치라는 방안을 고안하여 최적화된 MuTect를 구현하였다. 실제 유전체 데이터에 대해 실험한 결과, 제안한 최적화된 MuTect는 MuTect의 탐지 성능을 그대로 유지하면서 MuTect에 비해 수행 속도를 크게 향상시켰음을 보였다.

키워드 : 유전체 데이터 분석, 체세포 변이, 체세포 변이 위치 탐색, MuTect, 알고리즘 최적화
Key Words : Genome data analysis, Somatic mutations, Detection of somatic mutations, MuTect, Algorithm optimization

ABSTRACT

Techniques for detecting somatic mutations, which examine the causes of various diseases, especially the causes of tumor from genome data, play an important role in genome data analysis. In this paper, we investigated MuTect, one of the popular tools for detecting somatic mutations, and analyzed causes of its slow execution speed. This is due to inefficiency of MuTect algorithm where Pileup operator is executed redundantly and processing results of the operator is also duplicated. Thus, in order to solve the inefficiency, we devised a method called data rearrangement that changes data structure of genome data loaded on memory and implemented an optimized MuTect. We conducted experiments for real genome data, and showed that our optimized MuTect greatly improves the slow execution speed of MuTect without degradation of detection performance.

※ 본 연구는 2018년도 정부 (과학기술정보통신부, 보건복지부)의 재원으로 한국연구재단의 지원을 받은 보건의료기술연구개발사업 (HI15C3224)과 중견연구자지원사업 (2018R1A2B3001628), 두뇌한국21플러스사업의 지원을 받아 수행되었습니다.
◆ First Author : Seoul National University Department of Electrical and Computer Engineering, byunggook.na@gmail.com, 학생회원
° Corresponding Author : Seoul National University Department of Electrical and Computer Engineering, INMC, ASRI, ISRC and 공학연구원, sryoon@snu.ac.kr, 종신회원
논문번호 : 201901-424-C-RN, Received January 14, 2019; Revised March 4, 2019; Accepted March 12, 2019

Ⅰ. 서 론

체세포 변이 (somatic mutation)는 다양한 질병, 특히 암의 원인으로 알려져 있으며, 체세포 변이 판단을 위해 유전체 데이터를 적극 활용하는 중이다. 체세포 유전체를 분석하기 위해 Next Generation Sequencing (NGS) 이라는 고속 분석 방법을 통해 생물학적인 정보인 유전체를 컴퓨터로 분석하기 위한 텍스트 기반의 유전체 데이터[1]로 변환 저장한다. 본 논문에서 다루는 체세포 변이 위치 탐색은 방대한 양의 염기서열들로 이루어진 체세포 유전체 데이터를 분석하여 어느 위치에 변이가 일어났는지를 예측, 검출하는 기술이다. 체세포 변이 위치 탐색 기술은 유전체 데이터 분석 분야의 핵심 도구로써 깊이 연구되고 있다[1,2].

체세포 변이 위치 탐색을 위해 많이 쓰이는 알고리즘 중 하나인 MuTect[3]는 Broad Institute 사의 유전체 데이터 분석 플랫폼인 Genome Analysis ToolKit (GATK[4])의 체세포 변이 위치 탐색 알고리즘으로 제공된다. MuTect 프로그램을 실행하기 전에, 환자의 유전체 데이터의 염기서열들을 인간 참조 유전체 데이터 (human reference genome data, 이하 참조 데이터)에 정렬한다. MuTect 알고리즘은 정렬된 염기서열들을 참조 데이터의 각 위치별로 분할하고, 정보를 추출 및 가공하여 해당 위치의 변이 유무를 판별한다.

MuTect는 체세포 변이 판별 성능이 뛰어난 반면, 알고리즘 처리 속도가 매우 느리다는 단점이 있다. 본 논문에서는 MuTect 알고리즘을 검토하고, 느린 처리 속도의 원인이 되는 비효율적인 계산 처리 특징을 제시하고 이에 대해 심층적으로 분석하였다. 더 나아가, MuTect의 우수한 체세포 변이 판별 성능을 유지하면서, 알고리즘 수행 속도를 향상시키기 위해 데이터 재배치라는 기술을 고안하여 최적화 방안을 제안하였다. 데이터 재배치란 메모리 상에 로드된 유전체 데이터의 데이터 구조를 변환하는 것이며, 이를 통해 MuTect의 비효율성을 제거할 수 있다. 본 논문에서 제안한 알고리즘 및 구현 방법으로 사람의 전체 유전체 데이터를 처리할 때 기존 MuTect에 비해 약 130배 이상의 처리 속도 향상을 보였다.

1) NGS를 통해 얻은 유전체 데이터를 NGS 데이터라고도 하며, 이를 분석하는 과정을 NGS 데이터 분석 파이프라인이라고 한다. 변이 위치 판별은 NGS 데이터 분석 파이프라인의 대표적인 과정 중 하나이다.

Ⅱ. 본 론

2.1 MuTect 검토

MuTect가 분석하는 데이터는 NGS를 통해 얻을 수 있는 텍스트 기반의 유전체 데이터이다. 유전체를 이루는 염기인 아데닌, 사이토신, 구아닌, 티민을 텍스트 데이터로 나타내기 위해 각각 A, C, G, T로 문자로 표현한다. 유전체 데이터는 염기서열[2]의 집합이라 할 수 있으며, 각 염기서열은 텍스트 데이터의 관점에서 A, C, G, T로 이루어진 문자열이다.

MuTect 알고리즘은 인간의 참조 데이터를 정상인의 데이터로 가정한다. 참조 데이터를 참고하여, 분석하고자 하는 환자의 암 조직 유전체 데이터를 비교 분석한다. 하지만 정상 상태의 일란성 쌍둥이를 제외한 모든 사람의 유전체 정보는 상이하다. 즉, 참조 데이터와 다른 유전체 정보를 가지는 데이터 위치가 무조건 암의 원인으로 작용하는 것은 아니기 때문에, 환자의 정상 조직 유전체 데이터 또한 필요하다.

MuTect 알고리즘은 그림 1과 같이 참조 데이터의 염기 위치 순서대로 진행된다. 리드들은 참조 데이터의 위치에 맞추어 정렬 (mapping)되어 있는 상태다. Pileup 오퍼레이터를 통하여 각 위치에 해당하는 염기

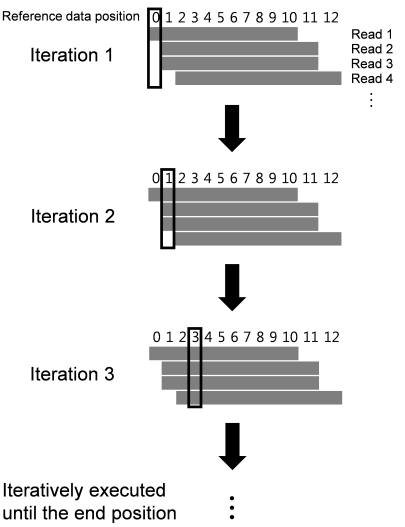

그림 1. 참조 데이터의 염기 위치 순서대로 처리되는 MuTect의 위치별 유전체 데이터 분석 방식
Fig. 1. Genome data analysis method of MuTect where the data is processed in order of base position of reference data

2) 흔히 Read라 하며, 본문에서는 리드라 명기하였다.

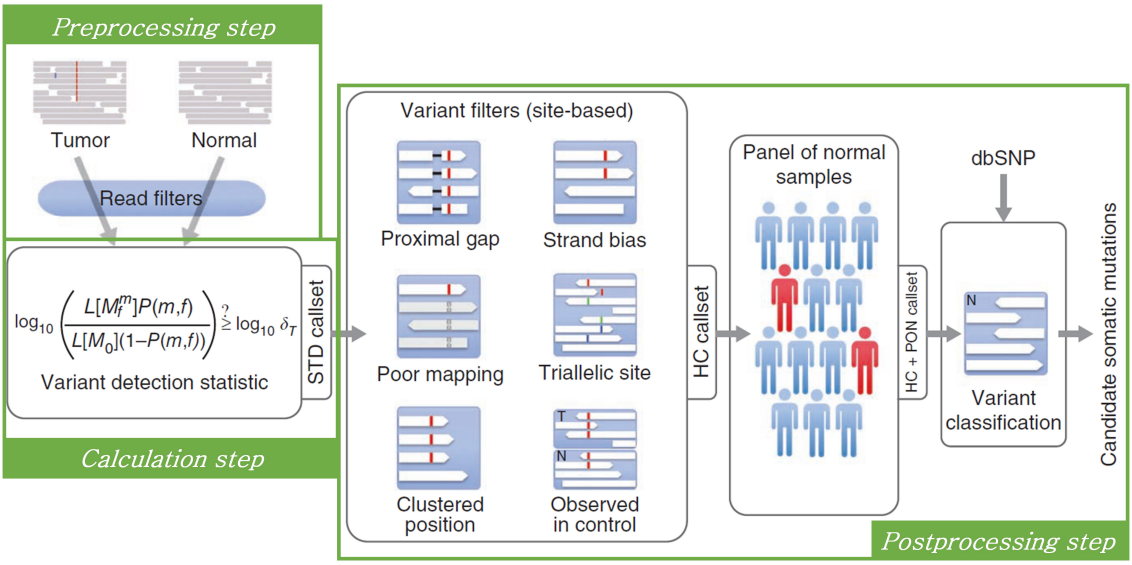

그림 2. MuTect 알고리즘 개요도 ([1]의 Figure 1을 발췌하여 수정)
Fig. 2. Overview of MuTect algorithm (modified after extracting Figure 1 in [1])

문자 및 정보를 리드 정보로부터 추출[3]하여, 추출한 정보들을 기반으로 해당 위치에 체세포 변이가 일어났는지를 판별한다. 판별을 위하여 각 위치 별로 세 작업이 진행된다 (그림 2): 전처리 작업 (Ⅱ.1.1절), 계산 작업 (Ⅱ.1.2절), 후처리 작업 (Ⅱ.1.3절). 전처리 작업에서는 계산 작업에 필요한 염기 정보만을 선별하고, 이 후 계산 작업에서 체세포 변이 판별 근거인 그림 2의 계산 작업 영역에 명시된 loglikelihood 기반의 점수를 계산하여 특정 임계점[4] 이상일 경우 변이가 발생하였다고 판단한다. 점수만을 가지고 판단된 변이는 실제로는 변이가 아닌 즉, 거짓 양성인 경우도 포함하고 있어, 거짓 양성으로 판단된 오류를 줄이기 위해, 각 위치별로 후처리 작업을 진행한 후 최종적으로 체세포 변이 유무를 판단한다.

2.1.1 전처리 작업

Pileup 오퍼레이터로부터 얻어진 리드들 중 리드의 정렬 품질, 신뢰도 등을 고려하여 계산 작업에 포함할 리드를 선별한 후 (그림 3-(a)), 리드를 이루는 염기들의 정보 또한 선별한다 (그림 3-(b)). 즉, 해당 위치에 정렬되어 있는 리드와 염기 정보 중 체세포 변이 판별에 적합한 정보만을 그림 3과 같이 리드와 염기 각각에 대한 조건 검사들을 적용하여 선별한다. 이 조건 검사들은 MuTect의 느린 처리 속도에 영향을 주는 중

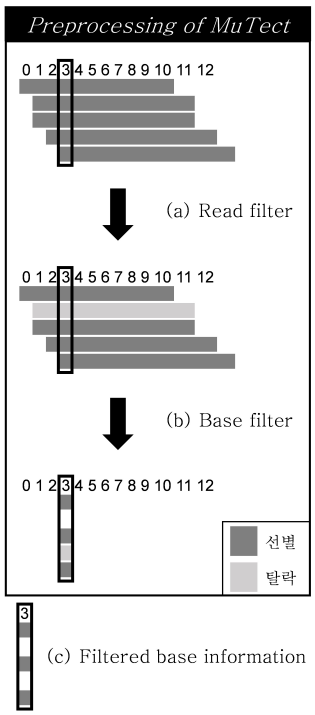

그림 3. MuTect의 전처리 작업: (a) 리드 정보 조건 검사, (b) 염기 정보 조건 검사, (c) 선별된 염기 정보 결과
Fig. 3. Preprocessing step of MuTect: (a) Read information filter process, (b) Base information filter process, (c) Filtered base information result

3) Pileup 오퍼레이터의 결과물은 리드 집합체이다. 그림 1.에서 각 위치별로 그려진 검은색 상자에 걸쳐진 리드들이 Pileup 오퍼레이터의 결과물이다.
4) MuTect에서는 6.3을 임계점 점수로 설정하였다.

요한 요소가 아니므로 본 논문에서 자세히 다루지 않으며, 각 조건에 대한 자세한 사항은 MuTect 보충서[5]에 명시되어 있다.

2.1.2 계산 작업

전처리 작업을 통해 선별된 염기 정보들 (그림 3-(c))만이 해당 위치에 대한 체세포 변이를 판별하기 위해 필요한 점수 계산에 활용된다. 각 위치에서 변이가 있는지에 대한 점수를 계산하기 위해, 참조 데이터의 염기와 다른 염기 셋을 후보 변이 염기로 상정한다. 예를 들어, 참조 데이터의 염기가 A라면 C, G, T를 후보 변이 염기로 간주하여 계산 작업을 수행한다. 후보 변이 염기에 대한 점수를 계산하기 위해서는 전처리 작업을 통해 선별된 염기와 염기 정보 중 염기 품질 (base quality)을 이용한다. 세 후보 변이 염기 중 가장 높은 점수를 가진 후보 변이 염기의 점수 (the log odds score, 이하 LOD 점수)가 6.3 이상이 되면 해당 위치에 체세포 변이가 '있을 수도 있다'라고 판단한다.

2.1.3 후처리 작업

LOD 점수가 6.3 이상이더라도 실제로는 변이가 아닐 수도 있는, 즉 거짓 양성 판단일 경우도 있다. 이는 유전체 데이터를 얻는 NGS 과정 중에 발생한 데이터 자체적인 오류, 혹은 리드를 정렬하는 알고리즘의 오류 (즉, 잘못된 정렬) 등과 같은 원인이 존재하기 때문이다[3]. 거짓 양성 판단을 최소화하기 위해 후처리 작업에서는 전처리 작업과 마찬가지로 Pileup 오퍼레이터를 활용하여 각 위치마다 리드 정보와 염기 정보들을 취합하여 거짓 양성의 가능성을 검토한다. 후처리 작업에서 진행되는 여러 필터는 생물학적 의미에 기반을 두고 있으며, 이에 대한 자세한 사항은 MuTect 본 논문[3]에 명시되어 있다.

2.2 MuTect 문제점 분석

2.2.1 Pileup 오퍼레이터 기반 구현 방식

우선, 전처리와 후처리 작업에서 중복적으로 호출되는 Pileup 오퍼레이터로 인한 비효율성 문제가 존재한다. 이를 설명하기 위해 길이가 L[5]인 리드가 있다고 가정하자. 이 리드는 위치 별로 호출되는 Pileup 오퍼레이터의 결과 (리드 집합체)에 L번 포함될 것이다. 시스템적으로 생각해볼 때, 이 리드 정보가 저장된 메모리에 접근 및 읽어오는 횟수가 L번 반복적으로 이루어지는 것이다. 심지어 후처리 작업에서는, 각 조건 검사할 때마다 Pileup 오퍼레이터를 새로 호출한다. 후처리 작업은 점수 계산 후 변이가 있을 수도 있다고 판단된 위치에서만 진행되지만, 이 중복 호출로 인한 오버헤드는 무시하지 못한다.

또한, Pileup 오퍼레이터의 중복 호출로 인하여 추가적으로 발생되는 비효율성이 존재한다. 리드에 대한 리드 정보 검사들도 Pileup 오퍼레이터에 의해 접근되는 수 (즉, 리드 길이)만큼 L번 중복적으로 수행된다. 후처리 작업의 필터들 역시 최대 L번까지 중복 수행될 가능성이 존재한다.

따라서 MuTect의 Pileup 오퍼레이터 기반 동작 방식에서는, Pileup 호출, 전처리 작업의 조건 검사들과 후처리 작업의 여러 필터들이 개별적으로는 짧은 수행 시간이 소요된다 하더라도, 이처럼 많이 누적된 오버헤드로 인한 영향은 전체 수행 시간에 크게 작용한다.

2.2.2 GATK 활용성 측면

MuTect는 Java 기반으로 제공되는 GATK 라이브러리를 사용하여 구현되었다. 프로그램 실행 속도를 높이기 위해서 기본적으로 고려되어야할 방안은 CPU, GPU 등의 멀티코어 자원을 활용한 병렬화 전략이다. 하지만, MuTect는 GATK에서 제공하는 병렬 프로그래밍이 가능한 특수 클래스를 상속하지 않은 채로 구현되어 있기 때문에 멀티코어 자원을 활용할 수 없다. 유전체 데이터 자체를 나누어서, 각 분할된 데이터에 대해 MuTect 프로그램을 따로 실행을 할 수 있는 방안도 있으나, 데이터를 분할하는 시간적, 저장 공간적 오버헤드가 존재하므로 좋은 방안이라 할 수 없다.

한편, 작은 요소들까지 클래스로 지원하는 GATK는 Java로 구현되었기 때문에, 이 작은 요소들이 모두 개별적인 소스 코드 파일로 이루어져 있다. 하지만 방대한 코드 양에 비해 이를 시스템 연산 자원에 맞추어 잘 활용할 수 있도록 도와주는 안내문서의 양이 적다. 따라서 GATK 아키텍처를 이해하는 것은 어려운 편이고, 이를 효율적으로 활용하는 것 역시 어렵다.

2.3 MuTect 최적화 방법

MuTect의 매우 긴 수행시간을 줄이기 위해서는 우선적으로 Pileup 오퍼레이터의 중복 호출로 인한 문제를 해결하는 것에 집중해야 한다. Pileup 오퍼레이터는 리드 단위로 메모리 상에 로드된 데이터로부터 유전체 데이터의 특정 위치에 위치하고 있는 리드들을

5) 본 논문에서 사용된 유전체 데이터는 모든 리드의 길이 101로 고정되어 있다.

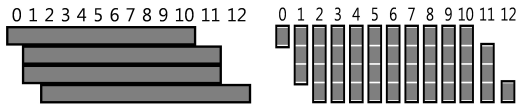

그림 4. 데이터 재배치: (a) 리드 단위 데이터 구조, (b) 위치 단위 데이터 구조 (데이터 재배치를 통해 얻은 데이터 포맷)
Fig. 4. Data rearrangement: (a) Read-wise data structure, (b) Position-wise data structure (otained by Data Rearrangement)

하나의 데이터 집합체로 추출하는 것이다. 본 논문에서는 이 중복적으로 수행되는 추출 작업으로 인한 비효율성이 큰 문제를 해결하기 위해 메모리 상에 로드되어 있는 데이터 구조를 변환하는 데이터 재배치라는 방안을 제안 및 구현하였다.

데이터 재배치란 그림 4와 같이 리드 단위 데이터 구조 (그림 4-(a))로 이루어진 유전체 데이터를 위치 단위 데이터 구조 (그림 4-(b))로 변환하는 작업을 의미한다. 데이터 구조 변환 시에 기존 MuTect의 전처리 작업과 후처리 작업에 필요한 정보들을 미리 계산 및 추출하기 때문에, Pileup 오퍼레이터를 한 번만 호출하는 것과 같은 효과를 가질 수 있다.

최적화된 MuTect 알고리즘 (그림 5)은 암 조직 유전체 데이터와 정상 조직 유전체 데이터를 각각의 데이터 재배치 작업을 통해 위치 단위 데이터 구조로 변환하고, 변환된 데이터를 이용하여 계산 작업과 후처리 작업을 진행한다. 후처리 작업 시에는 기존 MuTect와 달리 Pileup 오퍼레이터 호출과 여러 정보 추출 및 처리 작업이 생략된다. 이는 이미 데이터 재배치에서 수행되어 변환된 데이터에 저장되어 있기 때문이다. 이러한 정보들에는 리드 정보와 염기 정보, 참조 데이터와 비교·계산을 통해 가공된 정보 등이 포함된다.

2.3.1 데이터 재배치

그림 6은 데이터 재배치 작업에서 발생 가능한 경우들을 나누어 묘사한 것이다. MuTect의 전처리 작업과 마찬가지로 각 리드마다 리드 정보 조건 검사와 염기 정보 조건 검사들을 수행한다. 검사들을 모두 통과한 염기에 대해서만 계산 작업과 후처리 작업에서 필요한 정보를 추출 및 계산하여 염기 정보와 함께 해당하는 위치의 데이터 집합체 (collection)에 추가 (push)한다. 즉, 리드 입장에서 전처리 작업의 조건 검사를 1회만 한 후, 염기 단위로 정보를 쪼개어 각 위치에 해당하는 데이터 집합체에 재배치시키는 것이다.

계산 작업에서는 데이터 재배치를 통해 위치 별로

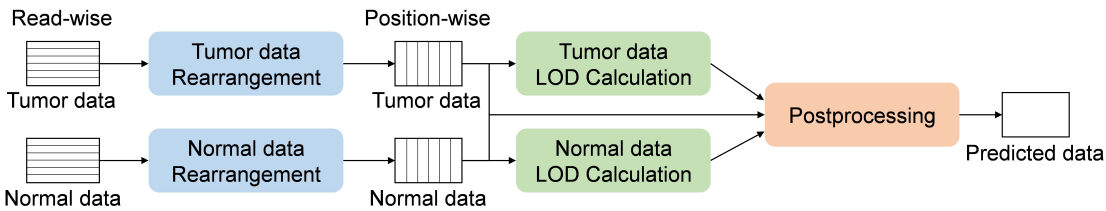

그림 5. 최적화된 MuTect 알고리즘 개요도
Fig. 5. Overview of Optimized MuTect algorithm

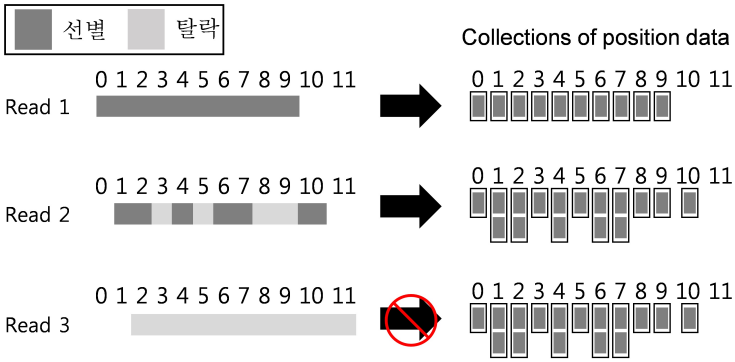

그림 6. 제안한 최적화된 MuTect의 데이터 재배치 작업: (Read 1) 모든 염기 정보가 선별된 경우, (Read 2) 일부 염기만 선별된 경우, (Read 3) 리드 자체가 선별되지 않은 경우
Fig. 6. Proposed data rearrangement step in optimized MuTect: (Read 1) the case that every base information is selected, (Read 2) the case that some of base information is selected, and (Read 3) the case that read information is not selected.

모인 염기 정보들을 활용하여 체세포 변이를 판별하기 위해, MuTect와 같은 방식의 LOD 점수 계산을 수행한다. 이러한 점수 계산은 위치 별로 저장된 염기 정보만을 활용하기 때문에, 각각 독립적으로 수행 가능하다. 이는 병렬화 용이성이 크다는 것을 의미하며, 최적화된 MuTect 알고리즘에 적합한 병렬화 전략에 대한 연구는 추가 연구로써 진행 중이다.

2.3.2 C/C++ 기반 구현

GATK는 유전체 데이터 분석 파이프라인 대부분에서 활용될 수 있도록 범용성에 초점이 맞추어진 라이브러리이기 때문에, GATK를 활용해서 구현하기에는 방대한 코드 양을 이해해야 한다는 어려움이 있다. 따라서 본 연구에서는 유전체 데이터를 사용할 수 있게 해주는 (즉, 유전체 데이터 IO를 담당하는) C 기반 라이브러리인 HTSlib[6]를 활용하여 C/C++ 기반으로 유전자 변이 위치 탐색 프로그램을 구현하였다.

GATK의 종속성에서 벗어남으로써 다음과 같은 여러 장점들을 가진다. 먼저, Pileup 오퍼레이터를 통한 다양한 기능을 제공하는 GATK보다 유전자 변이 위치 탐색에 더욱 적합한 구현을 할 수 있었다. 또한, GATK에서 주로 사용되는 Pileup 오퍼레이터 기반의 구현 방식이 아닌, 데이터 재배치라는 작업을 직접 구현하여 새로운 데이터 접근 방식을 제공한다. 마지막으로, Java 보다 시스템을 더 효율적으로 활용할 수 있는 C/C++ 언어를 이용하였기 때문에 보다 쉽게 병렬 프로그래밍 방식으로 확장할 수 있다. 멀티 코어 CPU나 GPGPU를 활용하는 방안은 앞서 언급한대로 본 연구의 추후 연구 주제로써 연구 중에 있다.

Ⅲ. 실 험

3.1 실험 방법 및 실험 환경

실험은 암 조직과 정상 조직으로부터 얻은 DNA 유전체 데이터에 대하여 진행하였다. 실제 환자의 전체 유전체 데이터 (whole genome sequence data, WGS)와 이 데이터로부터 추출하여 테스트용으로 사용한 염색체 1번 데이터 (chromosome 1 data, chr1)에 대한 수치정보는 표 1에 명시하였다.

표 1. 실험 데이터에 대한 수치 정보
Table 1. Numerical information of experimental data

Data	Size (GB)		Reference data length (= # of positions)
	Tumor	Normal	
chr1	3.9	3.9	249,250,621
WGS	127.8	133.7	3,095,677,412

기존 GATK 기반 MuTect와 C/C++ 기반 최적화된 MuTect의 번이 위치 탐색 성능은 동일하다. 본 논문에서 집중한 MuTect의 느린 프로그램 수행 속도를 개선한 효과를 보이기 위해 프로그램 실행 시간 비교 실험을 진행하였다.

실험은 Intel Xeon E5-2697 (2.7㎓) CPU와 768 GB 메모리가 탑재된 서버 상에서 수행되었다. C/C++ 코드 컴파일은 g++ 4.8.4 버전으로 하였으며, Java 코드 실행은 1.7.0_181 버전으로 하였다.

3.2 MuTect와 비교 실험

비교 실험 결과는 표 2에 정리하였다. 작은 크기의 chr1 데이터에 대해서 MuTect는 약 7일 (대략 10,080 시간), 제안한 방법은 약 64분을 소요하여 MuTect에 비해 약 157배의 속도 향상을 보였다. WGS에 대해서는 제안한 방법은 약 22시간 정도가 소요되었으나, MuTect에 대해서는 실험을 완료하지 못하여 프로그램 실행 시간을 포함시키지 못하였다. MuTect는 프로그램의 남은 수행 시간을 예상치로 출력하는데, WGS 데이터에 대해서는 하루 정도 수행된 뒤, 출력된 남은 시간 예상치가 약 17주였기 때문에 중단하였다. 실험 중에도 이 예상치는 조금씩 증가하는 모습을 보였기에 표 2에는 '17주 이상' 이라 명시하였다. 이를 120일로 가정한다면 제안한 방법에 비해 약 130배 정도 느리다고 볼 수 있다.

표 2. 프로그램 실행 시간 비교
Table 2. Runtime comparison for programs

Data	MuTect (Java)	Proposed method (C/C++)	MuTect2 (Java)
chr1	7 days	64 min.	70 min.
WGS	more than 17 weeks	22 hours 10 min.	18 hours 45 min.

3.3 MuTect2[7]와 비교 실험

WGS 데이터에 대해 MuTect 실험을 진행하지 못하여, GATK에서 제공하는 MuTect를 개량한 MuTect2로 추가 실험을 진행하였으며 이에 대한 결과를 표 2에 추가하였다.

제안한 방법과 비교했을 시, 작은 데이터인 chr1 데이터에 대해서는 제안한 방법이 미세하게 빨랐으나, WGS 데이터에 대해서는 조금 느렸다. MuTect2에 대한 자세한 분석은 추후 연구로 남겨두며, MuTect2는

MuTect와 마찬가지로 GATK 기반으로써 병렬화 확장이 쉽지 않다는 단점이 있다. MuTect2의 일부 단계에서만 병렬화가 적용되어 있으므로 병렬화 효과 또한 제한적이라 볼 수 있다.

3.4 추가 최적화 방법 시도

MuTect의 계산 작업에서는 모든 염기 정보에 대하여, 염기 품질을 이용해 loglikelihood와 pow 계산 등을 수행한다. 이 계산들의 수행시간을 줄이고자, 염기 품질 값의 범위가 [0, 60][8]임을 이용해, 미리 [0, 60] 범위에 해당하는 계산 값들을 계산하여 저장하였다. 이 후 LOD 계산 작업에서, 실제 계산을 하지 않고 저장된 값들을 이용함으로써 계산 오버헤드를 줄이고자 하였다.

chr1 데이터에 대해 실험한 결과, 최적화된 MuTect에 이 방안을 적용하기 전과 후의 수행 시간 차이는 4~5초 정도로 거의 없었다. 최적화된 MuTect의 계산 작업은 전체 수행시간 중 약 17% 정도였으며, loglikelihood와 pow 계산 등에 소요된 시간은 이보다 적을 것이기 때문에 이 절에서 시도한 최적화 방안은 효과가 미미했음을 추측할 수 있다.

Ⅳ. 결 론

본 논문에서는 체세포 변이 위치 탐색 기술의 대표 도구인 MuTect의 특성을 분석한 후, 발견한 비효율성을 개선하는 데이터 재배치라는 방안을 제안하고 실험적으로 MuTect의 비효율성을 개선하는 효과를 검증하였다. 기존 MuTect에 비해 chr1 데이터에 대하여 약 157배의 괄목할 만한 속도 향상이 있었다. WGS 데이터를 이용한 실험에서는 MuTect를 프로그램 종료까지 실행하지 못하였기 때문에 MuTect2에 대한 추가 실험을 수행하였다. 속도 측면에서는 비슷하였으나, 본 논문에서 제안한 방법이 병렬화 용이성이 더 높다는 장점이 있다. 추후 연구로써 최적화된 MuTect에 병렬화가 더 효율적으로 적용될 수 있도록 추가 개량 중이다. 본 연구를 통하여 유전체 데이터 분석을 통한 환자의 암 진단에 필요한 시간을 크게 줄일 수 있을 것으로 예상한다.

References

[1] C. Kwon, et al., "NGSOne: Cloud-based NGS data analysis tool," *J. Platform Technol.*, vol. 6, no. 4, pp. 87-95, Dec. 2018.

[2] J. Rhee and S. Shin, "Computational approaches for reconstruction of single individual haplotypes," *Commun. the Korean Inst. Inf. Scientists and Engineers*, vol. 32, no. 3, pp. 33-40, Mar. 2014.

[3] K. Cibulskis, et al., "Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples," *Nature Biotechnol.*, vol. 31, pp. 213-219, Feb. 2013.

[4] A. McKenna, et al., "The genome analysis toolkit: A mapReduce framework for analyzing next-generation DNA sequencing data," *Genome Res.*, vol. 20, no. 9, pp. 1297-1303, Sep. 2010.

[5] K. Cibulskis, et al., *MuTect supplementary* (2013), Retrieved Jan. 10, 2019, from https://media.nature.com/original/nature-assets/nbt/journal/v31/n3/extref/nbt.2514-S1.pdf.

[6] HTSlib: *C library for high-throughput sequencing data formats* (2018), Retrieved Jan. 4, 2018, from https://github.com/samtools/htslib.

[7] Broad Institute, *MuTect2* (2017), Retrieved Oct. 10, 2018, from https://software.broadinstitute.org/ gatk/documentation/tooldocs/3.8-0/org_broadinstitute_gatk_tools_walkers_cancer_m2_MuTect2.php.

[8] Wikipedia, *Phred quality score* (2019), Retrieved Jan. 11, 2019, from https://en.wikipedia.org/wiki/ Phred_quality_score.

나 병 국 (Byunggook Na)

2014년 2월 : 서울대학교 전기정보공학부 졸업

2014년 3월~현재 : 서울대학교 전기정보공학부 석박사통합과정

<관심분야> 생물정보학, 병렬 알고리즘

[ORCID:0000-0002-0459-4921]

윤 성 로 (Sungroh Yoon)

1996년 2월 : 서울대학교 전기공학부 졸업
2002년 6월 : Stanford University Electrical Engineering 석사 졸업
2006년 1월 : Stanford University Electrical Engineering 박사 졸업
<관심분야> 인공지능, 딥러닝, 생물정보학
[ORCID:0000-0002-2367-197X]