유전자 분석 어플리케이션의 시스템적 성능 특성 분석

박보영 ○ 구건재

고려대학교

boyoung0@korea.ac.kr, gunjaekoo@korea.ac.kr

Performance Analysis of the Modern Genome Alignment Application

Boyoung Park○ Gunjae Koo

Korea University

요 약

차세대 시퀀싱 기술의 개발로 유전체 데이터 크기가 매우 빠르게 증가하고 있다. 이에 따라 염기 서열 정렬이 새로운 빅데이터 워크로드로서 소프트웨어, 하드웨어적 가속 연구가 활발히 진행되고 있다. 본 연구에서는 이전의 매핑 프로그램보다 최소 2배 이상 빠른 Minimap2 프로그램을 분석했다. 그 결과, 병렬 처리 성능이 쓰레드 수에 비례할 정도로 효과적이나 메모리 데이터 베이스 공간의 한계로 메모리 용량이 충분하지 않을 때 최대 92.2배로 I/O 접근이 크게 늘어남을 확인했다. 그러므로 처리량 향상을 위해 개선된 시스템에서는 메모리가 부족하지 않게 수백 GB의 유전체 데이터를 관리할 수 있어야한다

1. 서 론

유전자 분석을 위한 어플리케이션은 방대한 길이의 유전자 염기 서열을 분석하기 위하여 고성능의 컴퓨팅 시스템과 효율적인 분석 알고리즘을 필요로 한다. 현대의 유전자 분석은 절편화, 시퀀싱, 리드 매핑(염기 서열 정렬), 변이 검출 단계를 통하여 분석을 진행한다. 이중 시퀀싱 단계는 NGS (Next Generation Sequencing) 기술로 인하여 성능이 대폭 향상되고 시퀀싱에 필요한 비용이 급격하게 줄어들었다. 이에 따라 현대의 유전자 분석에서는 시퀀싱의 다음 단계인 리드 매핑 단계에서 매우 긴 처리 시간이 소요되기 때문에 리드 매핑 단계가 전체의 유전자 분석 시스템의 성능을 좌우하고 있다. 연구자들은 리드 매핑의 성능을 개선하기 위한 하드웨어적인 연구([1]—[2])와 효율적인 근사 문자열 매칭을 위한 알고리즘 연구[3]를 진행해 왔다. 이 중 Minimap2는 리드 매핑 단계에서 범용적으로 사용하는 어플리케이션으로서 이전의 매핑 도구에 비해 높은 정확성과 빠른 처리 성능을 제공하고 있다[4].

본 연구에서는 유전자 분석의 리드 매핑에서 널리 사용되는 Minimap2의 처리 단계를 분석하고, 범용 컴퓨터 시스템에서의 성능 분석을 진행한다. 이러한 분석을 통하여 Minimap2의 높은 메모리 점유 특성과 데이터 I/O 시간을 보여주고 이를 개선하기 위한 연구 방향을 제시한다.

2. 배 경

Minimap2는 위에서 열거한 유전자 분석 단계 중에서 리드 매핑 단계에 사용하는 어플리케이션이다. Minimap2는 시퀀싱 단계의 결과 문자열인 리드(Read)를 참조 유전자 서열에 대해 정렬하거나, 서로 다른 두 리드에서 오버랩되는 부분을 찾아내어 드노보 어셈블리(de novo assembly)에 활용할 수 있다. Minimap2는 다른 유전자 정렬 방식과 마찬가지로 *seed-chain-align* 방식으로 리드 매핑을 진행한다. Minimap2의 각 단계는 아래와 같다.

Indexing: 일정 길이의 염기 서열 문자열을 대표하는 미니마이저(minimizer)를 생성하고 이를 해시테이블에 인덱싱한다. Indexing 단계에서 key로는 미니마이저, 매핑되는 value로는 참조 유전자 배열에서 문자열의 위치를 표시하는 배열이다.

Seeding: 유전자 샘플에서 생성한 미니마이저를 쿼리(query)로 사용하여 참조 유전자의 미니마이저와 일치하는 앵커(anchor)를 결정하고 참조 유전자 데이터에서 위치 배열을 수집한다.

Chaining: 수집된 위치 정보를 이용하여 참조 유전체에서의 위치를 기준으로 거리 상으로 가까운 앵커를 연결하여 근사적으로 매핑을 수행한다.

Aligning: 다이나믹 프로그램을 사용하여 최적으로 매핑이 될 수 있는 서열로 유전자 정렬을 수행한다.

Minimap2는 위에서 설명한대로 미니마이저를 사용하는 데이터 구조를 사용하여 최적으로 매핑될 수 있는 서열을 근사적으로 찾아내어 빠르게 유전자를 정렬할 수 있다. 이를 위하여 Minimap2는 참조 유전자 배열 및 쿼리로 사용하는 유전자 배열을 모두 호스트의 메모리에 저장하고 이 데이터 구조들을 반복적으로 접근하여 위치 정보를 수집하게 된다. 그러므로 Minimap2는 대용량(수십 GB)의 메모리 공간을 요구하며 위치 정보를 이용한 대용량의 계산을

수행하기 위하여 병렬 처리를 이용할 수 있다.

3. 시스템적 성능 분석

이 연구에서 우리는 Minimap2의 시스템적 성능 특성을 분석하였다. 특성 분석에 이용한 실험 환경은 표1과 같다.

표 1. 실험 환경

CPU	AMD Ryzen 7 2700 (8 cores / 16 threads)
Memory	DDR4 64GB
HDD	WDC WD40EFZX-68A 4TB
OS / Kernel	LinuxMint 19.3 (Ubuntu 18.04) / Linux kernel 5.4.0

3.1 데이터 이동량 및 데이터 이동 시간

Minimap2에서 수행하는 유전자 염기 서열 매핑은 참조 유전자 염기 서열에 대해 쿼리 유전자 염기 서열을 순서대로 정렬하는 작업이다. 사람의 경우 참조 유전자 서열의 크기는 약 3GB이고, 쿼리로 사용하는 유전자 서열은 그 크기가 수십 GB에서 수백 GB에 이른다([5]—[6]). Minimap2는 indexing 단계에서 3GB의 참조 유전자 서열에서 약 7GB의 해시데이터를 생성한다. 이렇게 생성된 해시데이터는 호스트의 메인 메모리에 상주한다. 앞에서 설명한대로 Minimap2는 seeding 단계에서 쿼리 유전자 시퀀스에서 생성한 미니마이저를 해시테이블에 쿼리로 보내기 때문에 이 단계에서 매우 많은 양의 데이터 이동이 발생한다.

표 2. 시퀀싱 기술별쿼리 데이터에 따른 참조 데이터 이동

시퀀싱 기술별 쿼리 데이터	쿼리 [GB]	참조 데이터 이동량[GB]	데이터 이동 시간[%]
ONT ERR012625_1	6.2	359	14.15
PacBio CLR SRR795762_2	0.8	456	15.89
PacBio Hifi ERR060204_1	5.0	431	22.47

표2는 쿼리 데이터에 따른 Minimap2의 참조 데이터 이동량과 데이터 이동시간을 보여주고 있다. 참조 데이터 이동량은 쿼리 요청에 의해서 해시테이블로부터 이동한 위치 정보 배열의 총 데이터 양을 의미하며 데이터 이동 시간은 표1의 실험 환경에서 Minimap2를 실행했을 때에 총 실행시간 대비 데이터 이동 시간의 비율을 나타낸 것이다. 실험 결과에서 알 수 있듯이 seeding 단계에서 Minimap2는 매우 방대한 양의 데이터(수백 GB)를 참조 시퀀스에서 읽어오게 된다. 그렇기 때문에 데이터 이동에 많은 시간을 소비하게 된다. 실험 결과에 따르면 Minimap2는 데이터 이동에만 총 실행 시간의 최대 22%의 시간을 소비한다.

3.2 계산의 병렬 처리 능력

Minimap2는 유전자 시퀀스에서 문자열을 추출하는 indexing 단계와 백트래킹 및 다이나믹 프로그램을 이용한 정렬 단계 모두 대량의 데이터를 이용한 반복적인 계산을 요구한다. 해당하는 처리 부분은 서로 독립적인 위치 정보 배열을 이용하는 경우가 많기 때문에 병렬처리를 이용할 경우 성능 향상을 기대할 수 있다.

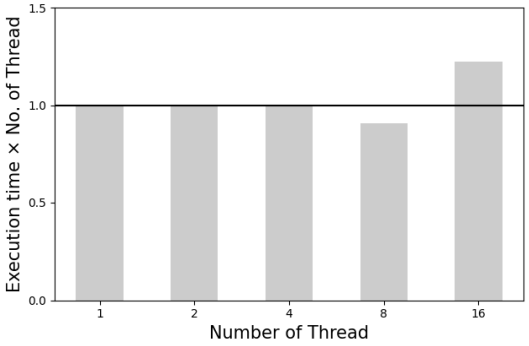

그림 1. 쓰레드 수에 따른 성능 향상 (기대 수행 시간)

그림1은 Minimap2의 계산 부분에서 생성하는 쓰레드 수에 따른 수행 시간을 보여준다. 각각의 경우의 수행시간은 쓰레드의 개수가 1일 때의 수행시간으로 normalize하였다. 즉, 쓰레드 개수가 2배가 될 경우 완벽하게 병렬처리가 가능하다면 수행시간은 1/2가 될 것이므로 Y축은 수행 시간에 쓰레드의 개수를 곱하고 이를 쓰레드가 1일 때의 수행시간으로 나누었다. 실험 결과에 따르면 쓰레드의 개수를 증가시킬 경우 Minimap2의 계산 시간은 그에 비례하여 감소하는 것을 알 수 있다. 쓰레드 개수가 16개일 경우 병렬화에 비해서 성능이 감소하였지만 이는 실험에 사용한 CPU가 8개의 physical 코어로 16개의 logical 쓰레드 실행을 지원하기 때문이다. 16개의 쓰레드를 이용할 때 성능이 감소하는 부분은 context switch에 따른 오버헤드 때문이다.

결과적으로 Minimap2의 계산 부분은 병렬처리를 이용할 경우 병렬화에 비례하여 성능 향상을 얻을 수 있다는 것을 알 수 있다. 그러므로 현재의 Minimap2를 CPU가 아닌 GPU나 가속기 구조를 사용한다면 계산 부분에서 높은 성능을 얻을 수 있을 것이라고 예상할 수 있다. 그렇지만, 앞에서 언급한 대로 Minimap2는 방대한 양의 데이터 이동을 요구하므로 병렬화를 시킬 경우 데이터 이동에 의한 오버헤드가 커질 것으로 예상할 수 있다.

3.3 메모리 요구 공간

Minimap2는 방대한 데이터 이동에 의한 성능 저하를 최소화하기 위하여 생성되는 참조 유전자 서열 및 해시테이블을 모두 호스트의 메인 메모리에 저장하여 사용하고 있다. 이는 필연적으로 매우 큰 크기의 메모리 공간을 요구하기 때문에 호스트 시스템의 비용을 증가시키는 요인으로 작용하며, 또한 메모리 공간의 제약으로 인해 많은 양의 유전자 분석을 불가능하게 하는 요인으로 작용할 수 있다.

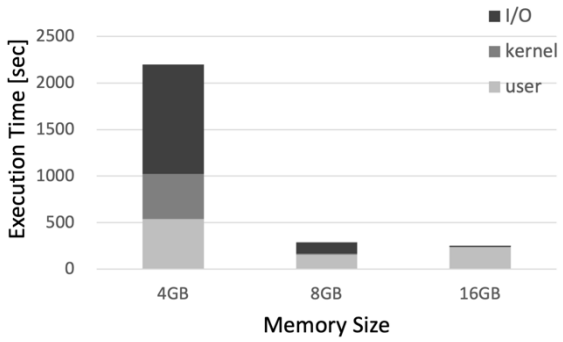

그림 2. 메모리 제한에 따른 사용자, 커널, I/O 시간

그림2는 메모리 공간 제한에 따른 Minimap2의 성능 변화를 보여주고 있다. 이 실험에서는 커널의 셋팅을 변경하여 사용할 수 있는 메모리의 크기를 제한하면서 Minimap2의 총 실행 시간 및 user time, kernel time, I/O time을 나누어서 측정하였다. Minimap2의 데이터를 모두 저장할 수 있는 크기인 16GB의 메모리 크기에 비해서 가용 메모리 용량을 8GB와 4GB로 제한하였을 때에 Minimap2의 실행시간은 각각 16.3배, 1.9배로 증가하였다. Kernel time과 I/O time을 합친 시스템 실행 시간은 각각 92.2배와 7.4배로 증가하였다. 이는 염기 서열 저장에 필요한 메모리 공간이 부족하여 page fault가 빈번하게 발생하고 저장장치로부터 필요한 데이터를 가지고 오는 데 많은 시간을 소요하기 때문이다.

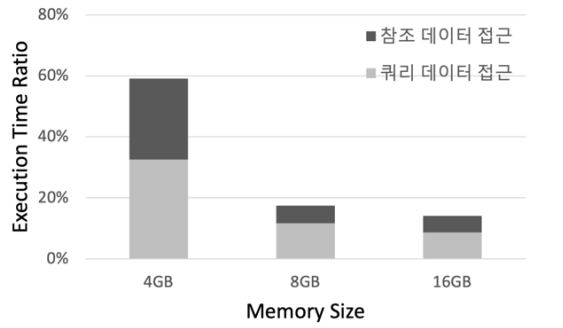

그림 3. 전체 실행 시간에서 쿼리 데이터와 참조 데이터의 이동 시간 비율

그림3은 Minimap2를 실행했을 때에 참조 염기 서열과 쿼리 염기 서열에 접근하는 시간의 비율을 나타낸 것이다. 가용 메모리의 크기가 작아지면서 전체 실행 시간에서 데이터 접근에 필요한 시간이 늘어나게 된다. 이에따라 데이터를 저장장치부터 가져오게 되면서 I/O 요청이 빈번하게 발생하게 된다. 메모리 공간이 줄어듦에 따라 쿼리 데이터에 비해 참조 데이터에 접근하는 시간이 크게 늘어난 이유는 7GB의 참조 데이터에 불규칙하게 접근하는 과정에서 I/O 요청이 크게 증가하기 때문이다. 이에 따라 CPU 사용율도 각각 3.8배, 1.3배로 감소하게 된다.

3.4 유전자 분석을 위한 시스템 연구 방향 제안

앞서 분석한 바와 같이 Minimap2가 수행하는 리드 매핑은 병렬적이 처리가 가능하면 처리 속도를 높이기 위하여 방대한 양의 메인 메모리 공간을 이용하고 있다. 유전자 분석 시스템의 효율을 높이기 위해서는 병렬처리를 극대화할 수 있는 GPU나 가속기 구조를 사용하는 방향을 제안할 수 있다. Minimap2의 계산 속도가 개선될 경우 데이터 이동에 필요한 시간이 주요한 오버헤드로 작용할 수 있으므로 이를 개선할 수 있는 연구가 필요하다. 또한 저장장치 공간을 사용하여 실행에 필요한 메모리 공간을 줄이는 연구가 필요하다.

4. 결 론

본 연구에서는 유전자 리드 매핑에 널리 사용되는 Minimap2의 성능을 분석하였고, 분석한 결과를 이용하여 향후 유전자 분석 시스템의 연구 방향에 대하여 제시하였다. Minimap2의 계산 부분 및 데이터 이동 부분의 분석을 통하여 계산 부분의 병렬화를 통한 가속이 가능함을 보였으며, 또한 대용량의 유전자 데이터를 저장하기 위한 메모리 저장 공간 문제와 데이터 이동에 따른 성능 오버헤드가 문제가 됨을 밝혀내었다.

5. 참고문헌

[1] Y. Turakhia, G. Bejerano, and W. J. Dally, “Darwin: A genomics co-processor provides up to 15,000 x acceleration on long read assembly,” *ACM SIGPLAN Notices*, 53, 2, 199-213, 2018.

[2] Y. Liu, Y., and B.Schmidt., “CUSHAW2-GPU: empowering faster gapped short-read alignment using GPU computing,” *IEEE Design & Test*, 31, 1, 31-39, 2013.

[3] M. Alser et al., “Technology dictates algorithms: Recent developments in read alignment,” *Genome biology*, 22, 1, 1-34.

[4] H. Li, “Minimap2: pairwise alignment for nucleotide sequences,” *Bioinformatics*, Vol. 34, 18, 3094-3100, 2018.

[5] D. L. Altshuler et al., “A map of human genome variation from population scale sequencing,” *Nature*, Vol. 467, 7319, 1061, 2010.

[6] NIH “Reference Genome GRCh38,” https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.39 (Published Feb.28, 2019)