그래프 기반 분산 시스템을 이용한 염기 서열 정렬

이준수*, 안재균*, 여윤구*, 노홍찬*, 박상현*
*연세대학교 컴퓨터과학과
e-mail:{solidpple, ajk, yyk, fallsmal, sanghyun}@cs.yonsei.ac.kr

DNA Sequence Alignment Using a Graph-based Distributed System

Jun-Su Lee*, Jae-Gyoon Ahn*, Yun-Ku Yeu*, Hong-Chan Roh*, Sang-Hyun Park*
*Dept. of Computer Science, Yonsei University

요 약

서열 정렬(sequence alignment)은 유전학(genomic)에서 널리 사용되는 도구 중 하나이다. 최근에는 차세대 시퀀싱 기술(NGS)이 발달함에 따라 데이터의 생산량이 크게 증가했고, 이에 따라 높은 처리량(throughput)을 가진 서열 정렬 알고리즘의 필요성이 증가하였다. 본 논문에서 제안하는 염기 서열 정렬 알고리즘은 시퀀스(sequence)데이터를 그래프 형태로 변형시킨 다음, 마이크로소프트사의 그래프 기반인 메모리(in-memory) 분산시스템(distributed system) 트리니티(Trinity)를 이용해 서열 정렬을 수행한다. 본 논문의 알고리즘은 트리니티 시스템에서 시뮬레이션 염기 데이터를 성공적으로 정렬하였으며, 슬레이브의 개수가 늘어날수록 빠른 속도를 나타내어 확장성(scalability)을 입증했다.

1. 서론

유전체(Genome)는 생명 현상의 정보를 담고 있는 가장 기본적인 물질이다. 유전체 내부의 유전자(gene)는 RNA와 단백질(protein) 형태를 거쳐 생명체를 구성하거나 기능을 수행한다. 또한 유전체는 조상으로부터의 유전 정보를 전달하고 다양한 원인으로 발생한 변이(polymorphism)를 누적함으로써 생물의 진화에서 중요한 역할을 수행하기도 한다.

2003년 게놈 프로젝트(Genome project) 이후 유전체 정보의 해석 기술은 급속도로 발전하였다. 해석 기술의 초창기에는 인간의 전체 유전체를 해석하는데 천문학적인 금액과 시간이 소모되었지만, 이른바 NGS(Next-Generation Sequencing) 기술이 급속하게 발전함에 따라 해석에 필요한 시간과 금액도 급격히 감소하였다. 이러한 유전 정보 해석 기술의 발전과 함께 염기서열 정렬(sequence alignment) 알고리즘의 중요성 역시 부각되고 있다. 염기서열 정렬 알고리즘은 유전체 재조립(genome resequencing), 유전체 변이의 탐색, TFBS(Transcription Factor Binding Site)의 탐색, 새롭게 발견된 유전자나 단백질의 기능 탐색 등 생물학 전반에서 널리 사용되는 알고리즘이다. 특히 NGS기술을 이용하면 해석하고자 하는 유전체의 염기 서열을 짧은 길이의 염기 서열인 리드(read) 형태로 대량 생산할 수 있으며, 이 리드를 염기서열 정렬 알고리즘을 이용해 참조 유전체(reference genome)에 정렬하고 차이점을 비교함으로써 해석하고자 하는 염기 서열을 빠르고 저렴하게 해석할 수 있다.

염기서열 정렬 알고리즘이 갖추어야 할 조건은 크게 처리량(throughput)과 정확도(accuracy)로 압축할 수 있다. 그 중에서도 NGS 기술이 발전함에 따라 리드의 생산량이 폭발적으로 증가하고 있기 때문에 이를 감당할 수 있는 높은 처리량을 갖춘 서열정렬 알고리즘이 필요하다. 또한 리드의 길이가 상대적으로 짧아(30~100개의 염기 서열) 리드가 갖고 있는 정보가 매우 제한적인데, 이런 짧은 염기 서열을 정확하게 정렬할 수 있는 정확도 역시 중요하다.

2. 관련연구

NGS 리드가 생산되기 시작한 이래 SOAP2[1], BWA[2], Bowtie[3] 등 여러 가지 염기서열 정렬 알고리즘들이 개발되었다. 그러나 진핵생물의 유전체

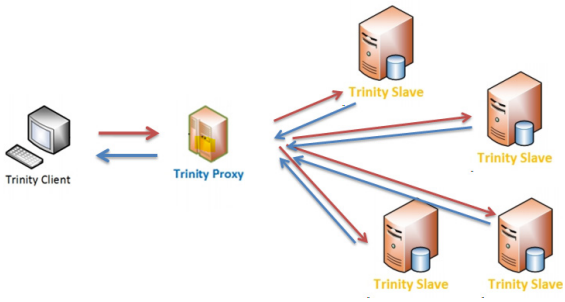
논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 도약연구지원사업 지원을 받아 수행된 것임(2012-010775)
또한, 본 연구는 정보통신산업진흥원의 IT/SW 창의연구과정의 연구결과로 지식경제부와 마이크로소프트에 의해 지원된 과제로 수행되었음
(NIPA-2012-H0503-12-1022)

에 광범위하게 분포하는 반복 서열(repeat)과 유전 과정과 돌연 변이로 발생하는 치환(substitution), 삽입(insertion), 결손(deletion). 중복(duplication)으로 인해 최적의 정렬 결과를 찾기 위해서는 많은 계산량이 필요하였다. 이 때문에 기존의 알고리즘들은 최적의 정렬 결과를 찾는 대신, 고유의 휴리스틱을 적용하여 높은 처리량을 획득하였다. 그러나 이러한 접근 방법은 필연적으로 정확도와의 트레이드오프(trade-off)를 갖게 되며, 분산 처리 기술의 발전 등 컴퓨팅파워가 계속 발전함에 따라 이러한 접근 방법을 반드시 따를 필요도 없게 되었다.

본 논문에서는 Microsoft사에서 개발한 그래프 분산처리 시스템인 트리니티[4,5]를 이용하여 염기서열 정렬 문제를 다룬다. 트리니티는 메모리 기반의 분산처리 시스템으로서, 그래프 형태의 데이터를 디스크에 기록하는 과정 없이 메모리 수준에서 해석하기 때문에 높은 처리 속도를 제공한다. 우리는 그래프 분산처리 기술을 적용하여 염기서열 정렬 문제를 해결하였다.

본 논문은 다음과 같은 구성으로 이루어져 있다. 3장에서는 트리니티 시스템에 대한 간략한 설명을 하고, 4장에서는 본 논문에서 제안하는 염기서열 정렬 알고리즘을 설명하고, 5장에서는 본 논문의 방법론을 검증하기 위한 실험 환경과 방법 및 결과, 6장에서는 결론, 마지막으로 7장에서는 추후 연구 방향에 대해서 설명한다.

3. 트리니티



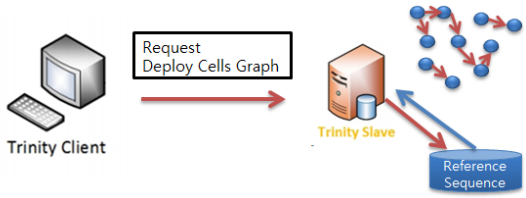
(그림 1) 트리니티 구조

(그림 1)과 같이 트리니티는 클라이언트, 프록시(Proxy), 슬레이브(Slave)로 구성 되어있다. 프록시는 슬레이브와 클라이언트 사이에서 메시지를 다루고 정보를 통합하는 역할을 한다. 슬레이브는 데이터의 일부를 저장하고, 프록시와 클라이언트로부터 받은 메시지를 처리하는 역할을 한다. 트리니티는 데이터를 메모리에 직접 저장하는 메모리기반 시스템으로서 데이터 접근 시간이 짧다는 장점이 있기 때문에 더 빠른 결과를 얻을 수 있다.

4. 분산 시스템을 이용한 그래프 서열 정렬

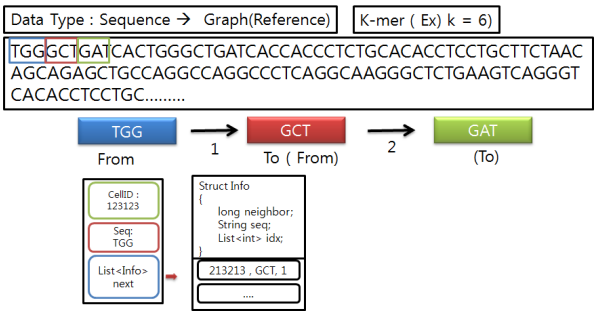
본 논문의 알고리즘은 크게 두 단계로 이루어져있다. 1) 그래프를 각 슬레이브에 배치(deploy). 2) 각 슬레이브로 리드를 쿼리(query)로서 전송하고 쿼리의 답을 프록시에서 종합해 클라이언트로 보내주는 단계이다.

4-1 참조시퀀스의 그래프 데이터 배치



(그림 2) 클라이언트 슬레이브에 요청

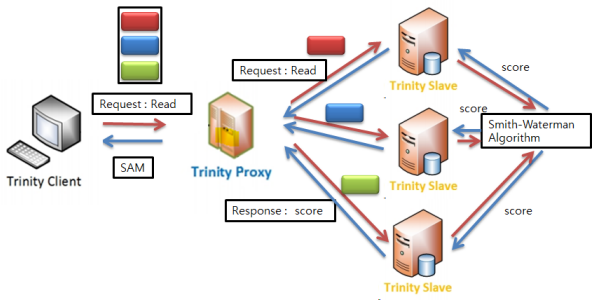
(그림 2)와 같이 클라이언트는 각 슬레이브에 참조 시퀀스를 그래프로 배치하도록 요청을 보낸다. 요청을 받은 각 슬레이브에서는 자신의 데이터 저장 공간에서 참조 시퀀스를 읽어 메모리에 올린다.



(그림 3) 시퀀스를 그래프 형태로 변환

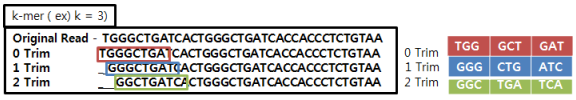
각 슬레이브는 로드한 참조 시퀀스 데이터를 (그림 3)과 같은 방법으로 k-mer(k는 파라미터 값 (ex k = 3))로 조각낸다. 다음으로 각 k-mer를 노드로 저장한 뒤, 참조 시퀀스 상에서 나타나는 k-mer 간의 from-to관계를 그래프의 간선으로 변환한다. 이 때 k의 값이 작아지면 노드의 개수가 적어지는 대신 간선의 개수가 많아진다. 이 경우 추후 쿼리를 종합하는 단계(그림 7참조)에서 계산량이 커지기 때문에 실험을 통해 적절한 k의 값을 찾는다.

4-2 그래프 쿼리



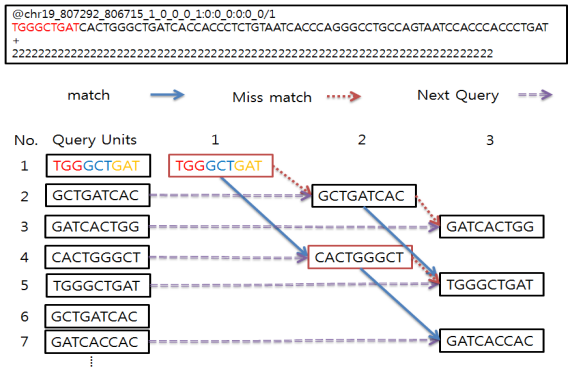
(그림 4) Proxy와 Slave간의 메시지 요청과 응답

(그림 4)는 그래프 쿼리에 대해서 클라이언트, 프록시, 슬레이브간의 메시지 요청과 응답에 대해서 나타냈다. 각 슬레이브에 그래프 데이터가 배치되면 클라이언트는 프록시에게 리드를 요청하고 프록시는 그 요청을 나누어 각 슬레이브로 전송한다.



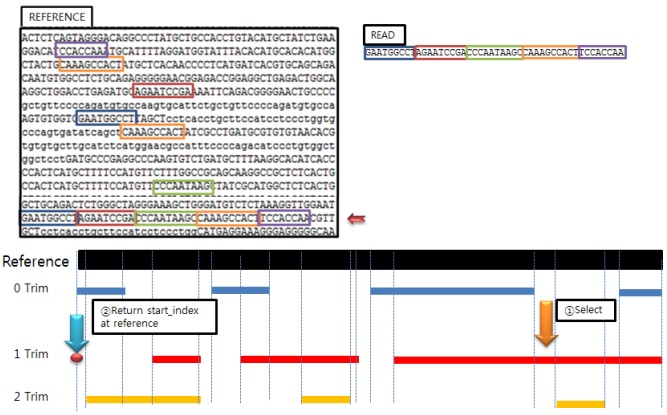
(그림 5) 트림 쿼리 순서

(그림 5)은 각 슬레이브로 들어온 리드 시퀀스를 참조 시퀀스와 동일하게 k-mer로 변환하는 과정을 나타낸다. 이 때, 참조 시퀀스의 어느 k-mer와도 정렬될 수 있도록 0, 1, 2, ..., k-1 bp씩 트림하여 총 k개의 세트를 생성한다.



(그림 6) 쿼리 순서

슬레이브에서는 각 트림 별로 그래프 리드를 k-mer로 변환한 뒤, 3hop을 정렬하는 k-mer를 기본 쿼리 단위로 서열 정렬을 수행한다. hop의 개수가 많아지면 많아질수록 정렬하는 시간은 늘어나지만 추후 쿼리를 종합하는 단계(그림 7참조)에서 계산량이 줄어들게 된다. 하지만 참조 시퀀스는 반복 서열이 많기 때문에 데이터 특성상 3hop정렬을 수행했다. 쿼리 중 첫 번째 노드의 정렬 위치를 탐색할 때에는 노드 아이디(Cell ID)(그림 3)를 기반으로 구축된 해시 인덱스를 이용하며, 이때의 탐색 속도는 O(1) 이다. 첫 번째 노드의 탐색이 성공하면 해당 노드에 저장되어 있는 이웃 정보와 다음 k-mer의 시퀀스와 비교한다. 이와 같은 방법(그림 6)으로 한 개의 쿼리에 해당하는 3개의 k-mer가 모두 매칭되면 중첩되지 않게 다음 쿼리를 준비하고, 매칭되지 않을 경우에는 일부 시퀀스의 중첩을 허용하며 다음 쿼리를 준비한다.



(그림 7) Align된 곳 중 최적의 위치 찾기

다음으로 참조 시퀀스에 정렬된 쿼리의 정렬 위치를 조합·연결하여 각 트림마다 참조 시퀀스에 연속적으로 매칭된 쿼리의 총 길이를 비교한다. 그 중 정렬된 염기 서열이 가장 긴 매칭을 최적의 위치로 선정하고 Smith-Waterman Algorithm[6]을 이용해 점수를 계산했다. 이렇게 각 슬레이브의 최적의 위치를 프록시에서 통합해 최종결과를 클라이언트로 전송하고 클라이언트에서는 SAM[7] 형태로 최종결과를 확인 할 수 있다.

5. 실험

실험을 위한 염기 서열 데이터는 Samtool[7]로 생성된 시뮬레이션 리드를 이용했다. 분산 처리 시스템의 성능을 활용하기 위해 필수적인 알고리즘의 확장성(scalability)과 ppv, sensitivity를 검증하기 위해 각각 슬레이브 개수와 k-mer의 수를 조절했다.

5.1 실험환경

(표 1) 실험환경

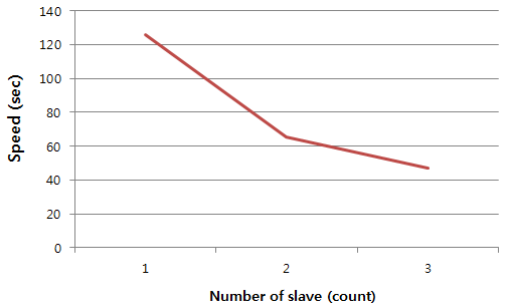
	OS	RAM	CPU
Proxy	Window 7(64bit)	8G	Intel i3-2120 3.30GHz
Slave	Window 7(64bit)	8G	Intel i5-3570 3.40GHz
	Window 7(64bit)	8G	Intel i5-3570 3.40GHz
	Window 7(64bit)	8G	Intel i5-3570 3.40GHz

5.2 실험 결과

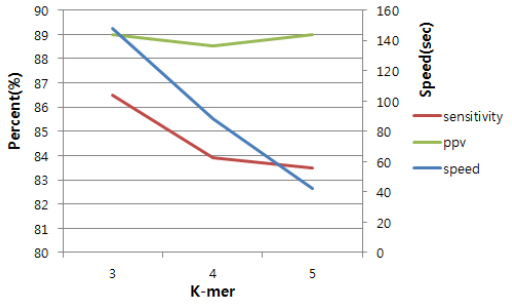
아래 공식에서 각각 mc는 정확히 맵핑된 리드 개수를 나타내고, mu는 맵핑되지 않는 리드 개수, mi는 부정확하게 맵핑된 리드 개수를 나타낸다. sensitivity(1)와 ppv(2)를 구하는 공식은 아래와 같다.

$$sensitivity = mc / (mc+mi+mu) \quad (1)$$

$$ppv = mc / mc+mi \quad (2)$$



(그림 8) 슬레이브 수에 따른 정렬속도 변화



(그림 9) k값의 변화에 따른 정렬 성능 변화

(그림 8)슬레이브의 개수가 많을수록 각 슬레이브에서 처리하는 양이 줄어들기 때문에 정렬 속도가 빨라지는 것을 볼 수 있다. 이는 본 논문의 알고리즘의 확장성을 검증했다. (그림 9)k-mer의 값이 늘어날수록 리드의 조각이 커지기 때문에 변이를 허용한 최적의 리드를 찾을 수 없어, sensitivity는 낮아지고, 반면 정렬속도는 빨라지는 것을 알 수 있다. 또한 ppv결과를 보고, 정확히 맵핑된 리드에 대해서는 정확도가 유지도 되는 것을 알 수 있다.

6. 결론

대용량의 NGS리드가 범용화 됨에 따라, 최적의 정렬 결과를 찾기 위해서는 많은 계산량이 필요하게 되었다. 본 논문에서는 기존 시퀀스 데이터를 일정 길이로 분할한 뒤, 분할된 염기 서열을 정점으로, 정점 간의 from-to 정보를 간선으로 갖는 그래프 형태로 변환하였으며, 이를 Microsoft사에서 개발한 그래프 분산처리 시스템인 트리니티에 적용하여 염기서열 정렬 문제를 다뤘다. 트리니티는 그래프 형태의 데이터를 디스크에 기록하는 과정 없이 메모리 수준에서 해석하기 때문에 높은 처리 속도를 제공할 수 있다. 시뮬레이션 리드를 이용하여 확장성을 검증하였으며, k-mer가 커질수록 더 빠른 결과를 얻을 수 있는 대신 확장성sensitivity가 낮아지는 것을 알 수 있다.

7. 추후연구

본 논문은 시퀀스 데이터를 그래프 형태로 변환해 그래프 기반 인 메모리 분산시스템에 접목하였으나, 아직 많은 부분에서 추후 연구가 필요하다. 참조 시퀀스에 전처리를 진행해 정렬비용을 줄이고, 정렬 시의 치환, 삽입, 결손. 중복 등을 고려해 더 최적의 해를 찾도록 개선할 예정이다.

참고문헌

[1] R. Li et al, “SOAP2: an improved ultrafast tool for short read alignment,” Bioinformatics Application note, vol. 25, no. 15, pp. 1966-1967, June 2009

[2] H. Li and R. Durbin, “Fast and accurate short read alignment with Burrows-Wheeler transform,” Bioinformatics, vol. 25, no. 14, pp. 1754-1760, May 2009

[3] B. Langmead, C. Trapnell, M. Pop, and S.L. Salzberg, “Ultrafast and memory-efficient alignment of short DNA sequences to the human genome,” Genome Biolo

[4] B Shao, H Wang, Y Li. The Trinity graph engine. Microsoft Research, 2012

[5] Trinity Manual by Microsoft Research Asia Copyright © 2012 Microsoft Corporation

[6] T.F. Smith and M.S. Waterman, "Identification of Common Molecular Subsequences," Journal of Molecular Biology vol. 147, pp. 195 - 197, 1981. gy, vol. 10, issue 3, Article R25, Mar. 2009

[7] H. Li et al, “The Sequence alignment/map (SAM) format and SAMtools,” Bioinformatics, vol. 25, no. 16, pp. 2078-9, June 2009