Journal of KIIT. Vol. 12, No. 7, pp. 145-154, July 31, 2014. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/kiitr.2014.12.7.145

분산 시스템에서 동작하는 서열 정렬 알고리즘 동향 분석

이준수*, 여윤구*, 노홍찬*, 윤영미**, 박상현*

A Survey of Sequence Alignment Algorithms with Distributed System

Jun-Su Lee*, Yun-Ku Yeu*, Hong-Chan Roh*, Young-Mi Yoon**, and Sang-Hyun Park*

이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2012R1A2A1A01010775).

요 약

최근 들어 차세대 시퀀싱(Next Generation Sequencing) 기술이 발전함에 따라 서열데이터의 생산량이 기하급수적으로 늘어나고 있다. 이렇게 급증한 차세대 시퀀싱 데이터를 처리하기 위해 많은 서열정렬 알고리즘이 등장했다. 하지만 많은 서열 데이터를 저장하고, 처리하는데 싱글 머신에서 동작하는 서열알고리즘은 한계가 있다. 이후 분산처리 시스템이 등장함에 따라 서열정렬 알고리즘은 분산에서 효율적으로 처리할 수 있게 되었다. 본 논문에서는 유전학(Genomics)에서 가장 널리 사용되고 있는 서열정렬 알고리즘을 사용하는 분산처리 시스템기반 알고리즘에 대해서 소개하고, 최근 등장한 새로운 분산처리 시스템과 생물정보학 관련 알고리즘의 특성을 분석해 기존 알고리즘의 문제점을 지적하고, 향후 연구 방향에 대해서 제시하였다.

Abstract

Recently, the production of sequence read data has exponentially increased as next generation sequencing (NGS) technology has been developed. A number of sequence alignment algorithms have been developed for handling NGS data. However sequence alignment algorithms based on single machine have limitation on saving and handling massive sequence data. Since distributed system have been introduced, sequence alignment algorithm can be effectively handled on the system. In this study, we describe various algorithms on distributed system using sequence alignment algorithms which are widely used in genomics. And we analyze characteristics of various distributed systems that have recently been developed and algorithm, related to bioinformatics. As a result, we point out the weakness of current algorithms with distributed system and suggest the directions for future research.

Keywords

next generation sequencing(NSG), sequence alignment algorithm, distributed system, in-memory cluster

* 연세대학교 컴퓨터과학과
** 가천대학교 컴퓨터공학과
· 접 수 일: 2014년 05월 19일
· 수정완료일: 2014년 07월 14일
· 게재확정일: 2014년 07월 17일

· Received: May 19, 2014, Revised: July 14, 2014, Accepted: July 17, 2014
· Corresponding Author: Sang-Hyun Park
Dept. of Computer Science, Yonsei University 50 Yonsei-ro, Sinchon-dong, Seodaemun-gu, Seoul 120-749, South Korea
Tel.: +82 2 2123-5714, Email: sanghyun@cs.yonsei.ac.kr

Ⅰ. 서 론

유전체(Genome)는 유전자(gene)와 염색체(chromosome)의 합성어로, 한 생물체가 생명 현상을 영위하는데 필요한 유전물질(DNA)의 집합체를 의미한다. 유전체 내부의 유전자(gene)는 RNA와 단백질(protein) 형태를 거쳐 생명체를 구성하거나 생명현상의 기능을 수행한다. 또한 유전체는 유전정보를 조상으로부터 전달하고, 전달 과정에서 생기는 변이(polymorphism)를 누적함으로써 생물의 진화에도 중요한 역할을 수행한다.

유전체 해석 기술은 초창기에는 천문학적인 금액과 시간이 소모되었지만, 2007년 차세대 시퀀싱(Next Generation Sequencing) 기술이 발전함에 따라 서열 데이터의 생산량이 기하급수적으로 증가했고, 그 결과 해석에 필요한 시간과 금액이 급격하게 감소했다. 특히 증가한 서열 데이터의 산출량은 정렬 알고리즘의 패러다임을 변화시키는 중요한 원인이 되었다. 일반적으로 하나의 유전체를 완전히 참조 조립하기 위해서는 전체 유전자의 수십 커버리지(coverage) 정도의 리드 서열을 생산한다. 인간의 유전체는 약 30억 bp를 기준으로 했을 때 이 경우 NGS 머신으로 생산되는 리드의 개수는 수십억 개에 달한다. 이처럼 유전체를 해석하는 기술의 발전과 함께 높은 처리량을 갖춘 서열 정렬(sequence alignment) 알고리즘의 중요성 역시 부각되었다.

서열정렬 알고리즘은 유전학에서 널리 사용되는 도구 중 하나로, 두 개의 서열 사이의 유사도를 이용해 구조적, 기능적 또는 진화적인 관계를 확인하는 도구를 말한다. 서열 정렬 알고리즘은 현재 다양한 분야에서 사용이 되고 있다. 하지만 초기에 등장한 서열 정렬 알고리즘은 싱글 머신 기반에서 동작하기 때문에 컴퓨팅 자원이 제한되어 있다. 그렇기 때문에 계속적으로 증가하는 서열 데이터를 저장하고, 처리하는데 한계가 존재한다.

2008년 Google에서 제안한 분산처리 시스템, Hadoop(https://hadoop.apache.org)이 등장함에 따라, 더 이상 싱글 머신에서 동작하는 서열 정렬 알고리즘이 아닌, Hadoop 기반 서열 알고리즘이 등장했다. Hadoop은 MapReduce[1] 프레임워크를 사용해 병렬적으로 빅 데이터를 처리하는 분산처리시스템이다. 분산처리 시스템은 여러 개의 머신을 사용해 병렬적으로 데이터를 저장하고, 처리하기 때문에 많은 데이터를 저장하고, 빠르게 처리할 수 있는 장점이 있다.

생물정보학(Bioinformatics)에서도 증가하는 바이오 데이터를 저장하고, 빠르게 처리하기 위해 다양한 분야에서 분산처리 시스템이 사용되고 있다. 그 중 많은 서열 데이터를 저장하고, 빠른 정렬 속도를 요구하는 서열 정렬 알고리즘을 사용하는 알고리즘은 CloudBurst[2], CloudAligner[3], CrossBow[4] 그리고 Myrna[5] 등이 다양하게 등장했다. 위에서 언급한 알고리즘들은 분산처리 시스템에서 동작하기 때문에 병렬적으로 많은 양의 서열 데이터를 저장하고, 효율적으로 처리할 수 있는 장점이 있다. 그러나 현재 Hadoop에서 동작하는 알고리즘은 Map Reduce를 사용해 병렬적으로 처리하기 때문에 중간 결과를 저장하는 단점이 있다. 결과적으로 많은 Disk I/O가 발생해 높은 처리량을 요구하는 서열 정렬 알고리즘이 사용되는 분야에는 적합하지 않다.

2013년 Microsoft에서 트리니티(Trinity)를 제안한 후, 기존에 널리 사용되고 있는 분산처리 시스템인 Hadoop의 단점을 보완할 수 있게 되었다. 트리니티는 in-memory 클러스터를 구축하기 때문에, Map Reduce 플랫폼을 사용하는 Hadoop과 다르게 데이터를 MapReduce에서 처리하기 위한 데이터 형태로 변환할 필요가 없을 뿐만 아니라 Map 단계와 Reduce 단계 사이에 중간 결과를 저장하지 않고, In-memory에서 처리하기 때문에 더 높은 처리량을 얻을 수 있다. 하지만 현재 서열정렬 알고리즘은 Hadoop에서 동작하는 알고리즘이 대부분이고, 트리니티에서는 동작하는 알고리즘이 없다. 그렇기 때문에 In-memory 분산 처리 시스템, 트리니티에서 동작하는 알고리즘의 개발이 필요하다. 트리니티는 서열 정렬 외에도 바이오 관련 빅 데이터에 모두 적용이 가능할 뿐만 아니라 높은 처리량도 얻을 수 있다.

본 논문은 차세대 시퀀싱 기술 등장 후, 서열 정렬 알고리즘의 흐름에 대해서 설명하고, Hadoop을 기반으로 사용되는 서열 정렬 알고리즘에 대해서 설명을 하고, 서열 정렬 알고리즘을 사용하여 처리되는 다양한 알고리즘에 대해서 설명한다. 마지막으로 서열 정렬 알고리즘의 특성을 확인하고, 현재 사

용되고 있는 Hadoop 메커니즘과 최근 공개된 트리니티[6] 메커니즘의 특성을 비교해 Hadoop의 문제점을 지적함으로써 추후 연구 방향에 대해 설명한다.

본 논문은 아래와 같이 구성되어 있다. 다음 장에는 서열 정렬 알고리즘과 분산처리 시스템, Hadoop에 대해 설명하고, 3장에서는 분산처리시스템에서 사용하는 서열 정렬 알고리즘에 대해 설명한다. 4장에서 새롭게 등장한 분산처리 시스템, 트리니티에 대해 설명한다. 마지막 5장에서는 현재 연구되고 있는 분산처리 시스템에서 사용되는 서열 정렬 알고리즘의 문제점에 대해서 설명하고, 추후 연구 방향에 대해서 제시한다.

II. 관련 연구

2.1 서열 정렬 알고리즘

서열 정렬(sequence alignment) 알고리즘은 유전학 연구에서 가장 널리 사용되는 도구 중 하나이다. 서열 정렬 알고리즘은 DNA, RNA 또는 단백질과 같은 두 개의 서열 사이를 비교해 진화적, 구조적 또는 진화적으로 유사한 지역(region)을 확인하는 알고리즘을 의미한다. 서열 정렬은 유전체 재조립(Genome re-sequencing), 유전체 변이 탐색(searching genome polymorphism), 그리고 Transcription Factor Binding Site(TFBS)의 탐색, 그리고 새롭게 발견된 유전자나 단백질의 기능 탐색 등 생물학 전반에서 널리 사용되는 알고리즘이다.

서열 정렬 알고리즘의 성능을 측정하는 값으로는 정확도(accuracy)와 처리량(throughput)으로 나눌 수 있다. 정확도는 참조 서열을 기준으로 리드 서열이 얼마나 정확하게 붙는지에 대한 수치를 의미하고, 처리량은 동일 시간 내에 얼마나 많은 리드 서열을 참조 서열에 정렬하는가에 대한 수치를 의미한다.

서열 알고리즘은 입력 값으로 DNA, RNA, 또는 단백질 두 개의 서열을 사용한다. 그 중 한 개의 서열은 참조 서열, 그 외 서열은 리드 서열로 사용한다. 서열 정렬의 결과 값으로는 리드 서열이 참조 서열에서 어디에 위치하는지에 대한 위치정보를 돌려준다. 예를 들어 그림 1은 서열 정렬 알고리즘에서 사용되는 입력, 출력 값을 나타낸 그림이다. 입력 값으로 사용되는 두 개의 서열 데이터 (참조 서열, 리드 서열)는 서열 정렬 알고리즘을 수행한 후에는 출력값으로 리드 서열이 참조 서열에 정렬된 위치 정보를 나타낸다. 여기서 오프셋(offset)은 리드 서열의 위치 인덱스를 의미한다. 빨간색 글씨(기울임)로 표시된 서열은 리드 서열이 정확하게 참조 서열 위치 (6~13)에 정렬된 것을 의미한다. 하지만 유전체는 조상으로부터 유전정보를 전달 받는 과정에서 생기는 변이(polymorphism)가 누적된다. 그렇기 때문에 리드 서열에는 치환(A→T), 삽입(ATC→AGTC), 그리고 삭제(ATC→AC)와 같은 변이가 존재한다. 그 외에도 참조 서열에 반복 서열(repeat)이 존재한다. 이처럼 변이와 반복서열 때문에 기존 싱글 머신에서 동작하는 서열 정렬 알고리즘은 최적의 정렬 결과를 찾는 대신, 고유의 휴리스틱을 적용하여 높은 처리량을 획득하였다.

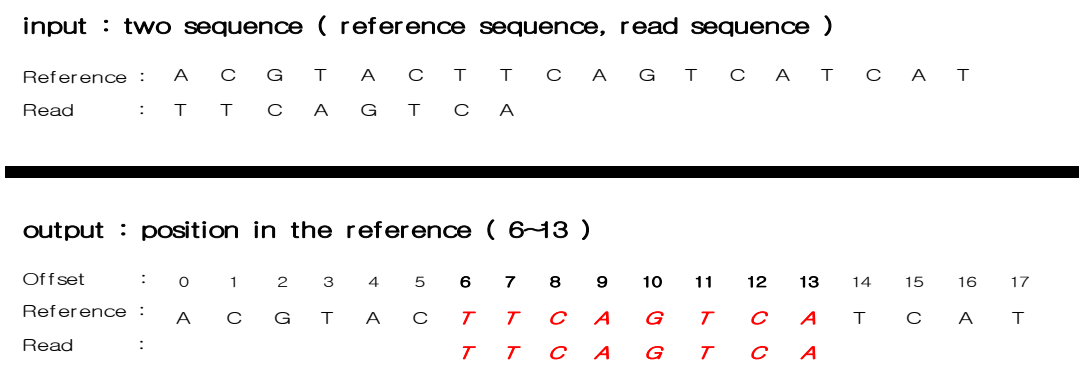

그림 1. 서열 정렬 알고리즘에서 사용되는 입력, 출력 값

Fig. 1. Input and output values which are used in the sequence alignment algorithm

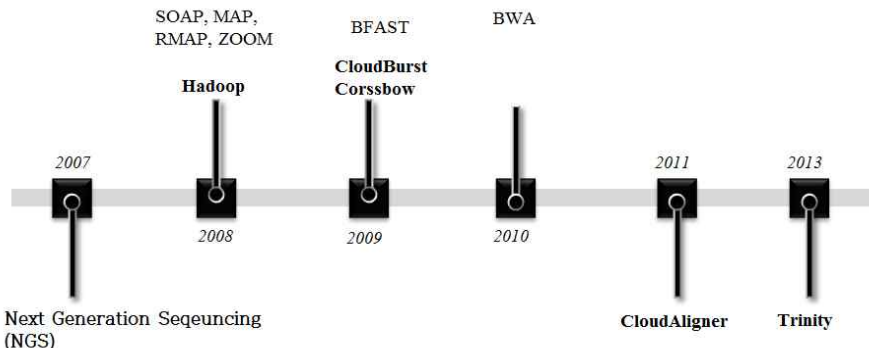

그림 2. 서열 알고리즘, 분산처리 시스템의 발전 흐름
Fig. 2. Flow of development of distributed processing system and sequence algorithm

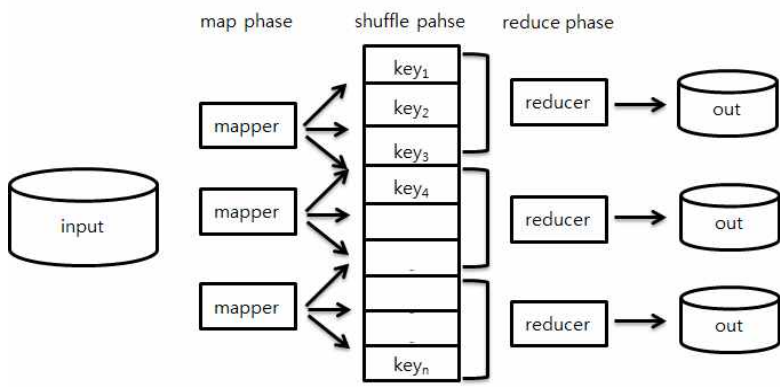

그림 3. MapReduce 개요
Fig. 3. MapReduce overview

그림 2는 서열 정렬 알고리즘의 시대적 흐름을 나타내는 그림이다. 2007년 차세대 시퀀싱(NGS) 기술이 발전한 후, 서열 데이터의 생산이 증가했다. 결과적으로 생산된 많은 서열을 처리하기 위해 서열 정렬 알고리즘이 필요했고, 2008년에 SOAP[7], RMAP[8][9], 2009년에 BFAST[10], 2010년에 BWA [11] 등 다양한 방법으로 정렬을 실시하는 서열 정렬 알고리즘이 등장하기 시작했다. 하지만 앞서 언급한 알고리즘은 싱글 머신에서 동작하는 서열 정렬 알고리즘으로 처리해야 하는 서열 데이터의 양이 많아짐에 따라 데이터를 저장하고, 처리하는데 한계를 갖고 있다. 2008년 Hadoop의 등장으로 싱글이 아닌 여러 대의 머신을 사용해 많은 서열 데이터를 병렬적으로 서열 정렬을 수행했다. 분산처리 시스템에서 동작하는 서열 정렬 알고리즘은 많은 서열 데이터를 저장하고, 효율적으로 처리할 수 있다. 그렇기 때문에 컴퓨팅 파워가 발전하고, 분산처리 시스템이 등장함에 따라 서열 정렬 알고리즘이 사용되는 알고리즘이 등장했다. 2009년 CloudBurst, CrossBow, 2010년 Myrna, 그리고 2011년 CloudAligner가 등장했다. 분산 처리 시스템에서 동작하는 알고리즘은 싱글 머신에서 처리할 수 없는 데이터의 양을 병렬적으로 처리했고, 증가되는 데이터를 간단하게 머신 하나를 추가해서 처리할 수 있는 확장성을 갖고 있다.

2.2. Hadoop 분산처리 시스템

Google에서 제안하고, 현재 가장 널리 사용되고 있는 MapReduce 프레임워크 기반 분산처리시스템 Hadoop은 테라바이트, 페타바이트 급의 빅 데이터를 병렬적으로 데이터를 저장하고, Map과 Reduce 단계로 나누어 효율적으로 처리한다.

그림 3은 Hadoop에서 MapReduce의 동작에 대해

서 간략하게 설명하고 있다. MapReduce는 map 단계와 reduce 단계로 구성되어 있다. 이 두 단계 사이에는 shuffle 단계로 map 단계에서 처리한 중간 결과를 저장한다. 최초 입력 파일들은 자동적으로 mapper의 개수에 따라 나누어 주고, 각 mapper는 Map 단계에서 입력 데이터로부터 사용자가 정의한 함수에 따라 key-value 쌍을 생성한다. shuffle 단계에서는 map 단계에서 계산된 key-value 쌍에서 key를 기준으로 값을 그룹화 한다. 이 과정을 통해 분산된 해시 테이블을 싱글 리스트로 만들어준다. 그 결과 사용자는 key 값을 이용해 value 값을 직접 접근하고, 얻을 수 있다. 마지막으로 reducer는 reduce 단계에서 유저가 정의한 함수에 따라 key와 연관된 value의 리스트를 계산하고 최종적으로 출력 파일들을 만들어준다. 이처럼 map과 reduce는 독립적으로 기능을 수행한다. 그렇기 때문에 사용자는 map과 reduce에서 수행하는 코드를 따로 구현해야 하는 번거로움이 있다. map과 reduce의 개수는 사용할 수 있는 프로세서의 코어의 개수에 따라 설정하고, 코어 개수가 많을수록 실행 시간이 줄어든다.

Hadoop에서 동작하는 컴포넌트로는 Sqoop[12], Pig[13], 그리고 Hive[14] 등 다양하게 존재한다. 하지만 위에서 언급한 컴포넌트들은 공통적으로 핵심 컴포넌트인 MapReduce를 사용하기 때문에 Map Reduce 프레임워크에서 동작할 수 있는 데이터 형태로 변형시켜야 하는 비용과 map과 reduce 사이에 중간 결과를 저장하기 때문에 많은 Disk I/O가 발생하는 한계가 갖고 있다. 현재 이러한 문제를 해결하기 위해 다양한 플랫폼들이 등장하고 있다.

III. 분산처리시스템에서 사용되는 서열정렬 알고리즘

차세대 시퀀싱 기술의 발전 이후, 생산된 많은 시퀀스를 처리하기 위해 분산처리시스템에서 동작하는 서열 알고리즘이 많이 등장하고 있다. 서열 정렬알고리즘은 Mapping, Whole genome assembly, 그리고 RNA-seq 분석 등 다양하게 사용된다.

3.1 Mapping

현재 분산처리 시스템에서 동작하는 Mapping 알고리즘은 CloudBurst와 CloudAligner가 존재한다. 두 알고리즘은 기존 싱글 머신에서 동작하는 서열 알고리즘과는 다르게 분산처리 시스템, Hadoop에서 MapReduce를 사용해 서열 정렬을 병렬적으로 실시하였다. 두 알고리즘은 최근 차세대 시퀀싱 기술의 발전으로 생산된 많은 서열 데이터를 저장하고, 효율적으로 처리하고 있다.

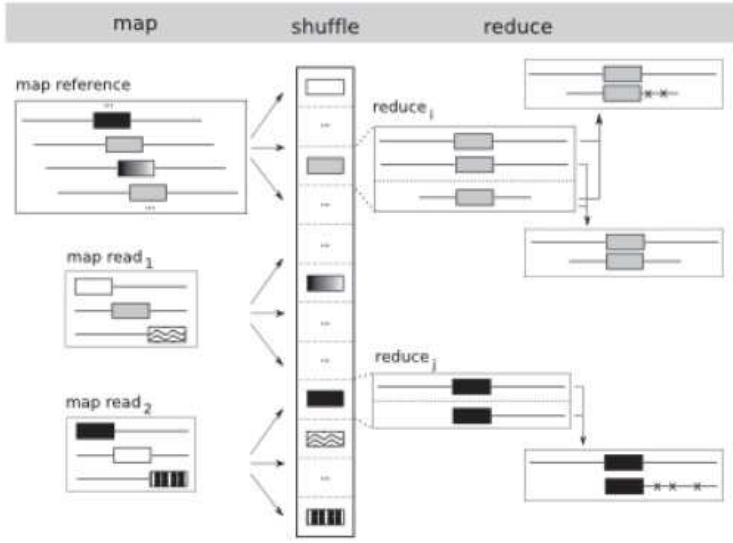

그림 4. CloudBurst 작업 흐름 [2]
Fig. 4. CloudBurst work flow [2]

3.1.1 CloudBurst

CloudBurst는 short read-mapping 프로그램으로 개발되었다. CloudBurst는 map, shuffle 그리고 reduce 단계를 통해 서열정렬을 실시한다. map 단계에서 참조 서열에서 모든 k-mer(길이가 k인 서열 조각)을 생성하고, shuffle 단계에서 리드 서열과 참조 서열 사이에 공유하는 k-mer를 그룹화를 실시한다. 마지막으로 reduce 단계에서 seed-and-extend 방법을 사용해 시드(정확하게 정렬된 서열)를 찾은 뒤, Landau Vishkin[15] 알고리즘을 사용해 시드를 확장시키고, k-differences를 허용하는 정렬 결과를 찾는다. CloudBurst는 파라미터 값에 따라서 유일하게 정렬된 리드 정보만을 보고하는 방식과, 모든 정렬 결과를 출력해준다. 하지만 이때 모든 정렬 결과를 출력해주면 매우 낮은 처리 속도(리드 1M개 했을 때, 56M개, 2M개 했을 때 113M개, 약 56배)를 보여준다. CloudBurst는 pair-end reads를 지원하지 않는다.

3.1.2 CloudAligner

CloudAligner는 long read-mapping 프로그램으로서 개발되었다. CloudAligner는 CloudBurst와 동일하게 seed-and-extend 방법을 사용해 서열 정렬을 실시한다. 하지만 CloudAligner는 CloudBurst와 다르게 기존 Map과 Reduce 두 단계를 모두 수행하는 것이 아닌, Reduce에서 하는 작업을 Map 단계에서 모두 처리한다. 결과적으로 Map과 Reduce 사이에 중간 결과를 저장하지 않기 때문에 Map과 Reduce를 모두 사용하는 알고리즘인 CloudBurst와 비교했을 때 더 높은 처리량을 보여준다. 또한 CloudBurst와 다르게 CloudAligner는 다양한 데이터 포맷을 지원한다. input으로는 fastq, fasta, output으로는 SAM, BED를 지원한다. 하지만 실제로는 아직 완벽하게 SAM과 fastq 포맷은 지원하지 못한다. 또한 CloudBurst와 다르게 pair-end-reads를 지원한다.

3.2 Whole Genome Re-sequencing

Whole Genome Re-sequencing은 현재까지 밝혀지지 않은 참조 서열이 있는 경우, 전체 게놈 조립(Whole Genome Assembly)의 비용을 절약하기 위해 선택했다. Whole Genome Re-sequencing은 참조 서열에 서열을 정렬해 SNP(단 염기 다양성)을 찾아서 전체 유전체를 생성한다. 얻어진 결과를 바탕으로 육종이나 질병에 대한 유전적 요소를 유추할 수 있다. 하지만 Whole Genome Re-sequencing을 수행하기 위해서는 많은 저장 공간과 높은 컴퓨팅 계산 량을 요구하기 때문에, CrossBow는 분산처리 시스템 Hadoop의 MapReduce를 사용해 병렬적으로 처리했다.

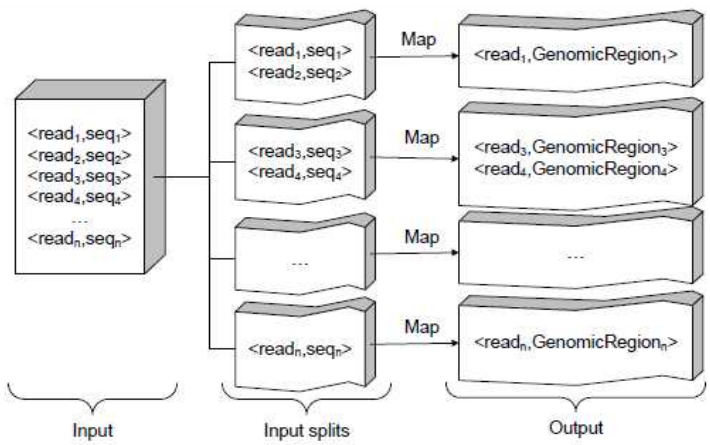

그림 5. CloudAligner 작업 흐름 [3]
Fig. 5. CloudAligner work flow [3]

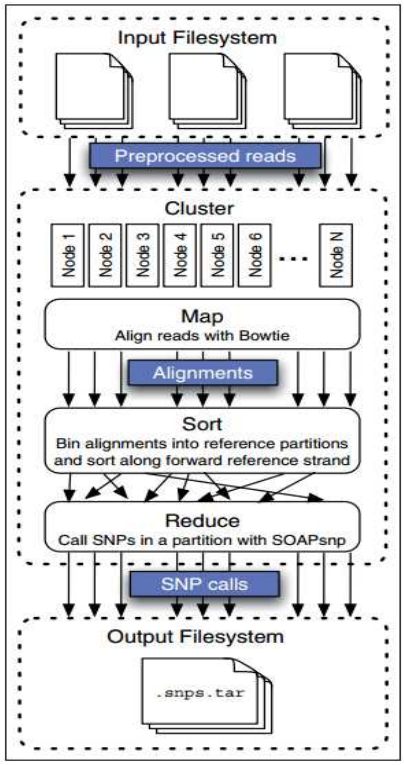

그림 6. CrossBow 작업 흐름 [4]
Fig. 6. CrossBow work flow [4]

CrossBow는 그림 6과 같이 Map 단계와 Reduce 단계를 거쳐 최종적인 결과를 보여준다. Map 단계에 들어가기 전에 MapReduce에서 처리할 수 있는 데이터 형태로 리드를 프리 프로세싱을 진행하고, Map 단계에서는 각 노드에서 기존에 개발된 서열 정렬 알고리즘, bowtie를 이용해 서열 정렬을 실시한다. 그 이후 중간 결과를 조건에 맞게 정렬한 뒤, Reduce 단계에서는 각 머신에서 SOAPsnp[16]을 사용해 SNP(단 염기 다양성)을 찾는다. 최종적으로 출력 파일들을 생성해준다.

3.3 RNA-seq 데이터 분석

Microarray 데이터는 생물정보학에서 흔하게 사용되는 데이터 중 하나이다. 연구자들은 Microarray에 있는 유전자의 발현량을 이용해 다양한 분야에서 사용하고 있다. 하지만 Microarray는 노이즈를 많이 포함하고 있기 때문에 정확한 값을 얻기에 어려움이 있다. 그렇기 때문에 최근 많은 연구자들은 RNA-seq를 사용해 유전자의 발현 값을 계산한다. 하지만 RNA-seq를 이용해 유전자의 발현 값을 얻는 과정은 계산량이 크기 때문에 Myrna 알고리즘은 분산처리 시스템 Hadoop의 MapReduce 환경에서 큰 RNA-seq 데이터 셋에서 기존 서열 정렬 도구, bowtie를 사용해 정렬을 실시하고, 정렬된 결과를 바탕으로 RPKM(reads per kilobase per million reads) 값으로 유전자의 발현 값을 계산한다. 이렇게 얻어진 유전자의 발현 값은 microarray에서의 유전자 발현 값 보다 더 정확한 값을 계산할 수 있고, microarray를 이용해 실험을 수행하는 모든 분야에 적용이 가능하기 때문에 다양한 분야에서 더 좋은 결과를 얻을 수 있다.

Ⅳ. 트리니티

트리니티는 Microsoft에서 제안한 In-memory 기반 분산처리 시스템이다. 그림 7은 트리니티는 아키텍처에 대해서 보여주고 있다. 트리니티 시스템은 다양한 컴포넌트들로 이루어져 있다. 컴포넌트들은 이미 구현되어 있는 API를 이용해 쉽게 메시지를 생성하고, 네트워크를 통해 통신한다. 트리니티는 하는 역할에 따라 크게 슬래이브, 클라이언트, 그리고 프록시로 나누어진다. 슬래이브는 실질적으로 처리하는 데이터를 저장하고, 유저가 정의한 주요한 핵심 알고리즘을 실행한다. 또한 프록시와 클라이언트로부터 네트워크를 통해 메시지를 주고받는다. 프록시는 슬래이브와 클라이언트 사이에 중간 중계자로써 슬래이브에서 받은 결과를 통합한 후 클라이언트에게 전송한다. 프록시의 역할은 클라이언트가 대신 수행할 수도 있기 때문에, 항상 트리니티 시스템에 포함될 필요는 없다. 마지막으로 클라이언트는 트리니티 시스템과 사용자(end-user) 사이에서 인터페이스 역할을 한다.

트리니티의 가장 큰 장점은 Hadoop과 다르게 MapReduce 프레임워크를 사용하지 않고, 최적화된 메모리 관리자와 네트워크 의사소통을 통해, 효율적인 병렬 컴퓨팅을 지원한다. 결과적으로 트리니티는 메모리 클라우드를 구축해서 in-memory 데이터를 빠르게 처리할 수 있다.

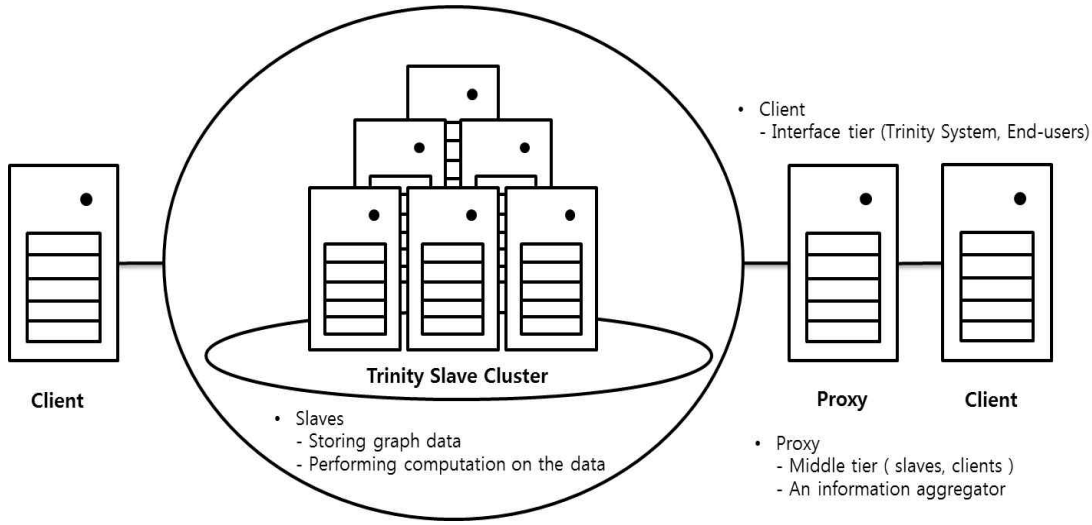

그림 7. 트리니티 시스템 아키텍처
Fig. 7. Trinity system architecture

트리니티는 그래프 형태의 데이터에서 그래프 탐색, 홉(hop) 탐색 등 그래프 형태의 데이터를 효율적으로 처리 할 수 있는 그래프 기반 분산 처리 시스템이다. 트리니티는 그래프에서 노드로서 사용되는 cell을 생성하고, CellID를 통해 해당 cell에 바로 접근이 가능하다. cell에는 노드의 이웃 정보 등 다양하게 저장이 가능하다. 트리니티는 API를 이용해 쉽게 노드를 생성하고, 접근이 가능하며, 각 컴포넌트 간에 통신이 쉬워 코딩하기에 간편한 장점이 있다. 또한 슬래이브의 개수를 늘림에 따라 더 빠른 처리 속도를 얻을 수 있는 분산처리시스템의 특성을 갖고 있다.

V. 분석 및 결론

유전학에서 서열 정렬 알고리즘은 기본이 되는 도구 중 하나이다. 서열 정렬 알고리즘은 차세대 시퀀싱 기술이 발전된 이후, 서열 데이터의 양이 늘어남에 따라 다양한 방법으로 처리하는 싱글 머신 기반 서열 정렬 알고리즘이 등장했다. 하지만 싱글 머신 기반의 알고리즘은 많은 서열을 저장하고, 처리하는데 한계가 존재한다.

최근 생물 정보학에서 사용하는 데이터의 크기가 점점 커지고 다양해지고 있다. 그렇기 때문에 방대하고 다양한 데이터를 저장하고, 처리하기 위해 Hadoop에서 동작하는 알고리즘이 등장하고 있다. 그 중 많은 저장 공간을 요구하고, 증가한 서열 데이터를 빠르게 처리하기 위한 서열 정렬 알고리즘을 사용하는 도구들이 많이 등장하고 있다.

Hadoop은 최근 널리 사용되는 분산처리 플랫폼으로 MapReduce를 사용한다. Hadoop의 등장 이후, 싱글기반 알고리즘의 문제를 해결하기 위해 여러 머신을 사용해 병렬적으로 처리 할 수 있는 알고리즘이 등장하기 시작했다. 최근 서열 정렬 알고리즘을 사용한 CloudBurst, CloudAligner, CrossBow 그리고 Myrna는 많은 데이터를 저장하고, 서열 정렬 알고리즘이 요구하는 높은 처리량과 정확도를 얻기 위해 Hadoop의 MapReduce를 사용했다.

그러나 MapReduce를 사용하는 Hadoop은 처리해야 하는 데이터를 MapReduce에서 처리할 수 있는 데이터의 형태로 변환을 해야 할 뿐만 아니라, Map과 Reduce 사이에 중간 결과를 저장해야 하기 때문에 많은 Disk I/O가 발생한다. 이러한 Disk I/O는 높은 처리량을 방해하는 요소가 된다. 또한 Hadoop은 테라바이트, 페타바이트 급의 데이터를 처리하는데 디자인되어 있기 때문에, 기가바이트 급의 데이터를 처리하는 서열 정렬 알고리즘에는 적합하지 않을 수 있다.

최근 Microsoft에서 제안한 트리니티는 Hadoop과 달리 MapReduce 프레임워크를 사용하지 않고, 데이터를 처리한다. 트리니티는 In-memory 클러스터를 구축하기 때문에 빠르게 데이터를 처리할 수 있다. 트리니티는 Hadoop과 다르게 MapReduce에서 처리할 수 있는 데이터의 형태로 변환할 필요가 없고, Map과 Reduce 단계 사이에 중간 결과를 저장할 필요 없이 연산을 수행한다. 그렇기 때문에 필요 없는 Disk I/O가 발생하지 않는다. 그 결과 트리니티는 Hadoop의 MapReduce에서 서열 데이터를 처리하는 것보다 훨씬 더 빠르게 처리 할 수 있다. 또한 최근 컴퓨팅 파워가 증가했기 때문에 메모리의 크기가 커지고, 서열 정렬을 수행하기 위해 충분한 크기가 되었다. 이처럼 서열 정렬 알고리즘은 In-memory 클러스터에서 처리하는 것이 더 효율적이다.

본 논문에서는 차세대 시퀀싱 기술의 등장을 시작으로 싱글 머신에서 동작하는 서열정렬 알고리즘의 특성을 설명하고, 분산처리 시스템에서 동작해야 하는 이유에 대해서 설명을 했다. 또한 현재 분산처리 시스템에서 다양하게 동작하는 서열 정렬 알고리즘의 활용, 중요성 그리고 한계에 대해서 설명을 했다. 최근 제안된 서열정렬 알고리즘은 대부분 Hadoop 분산처리 시스템에서 동작하고 있다. 하지만 최근 In-memory 클러스터를 구축해 데이터를 처리하는 분산처리 시스템 트리니티가 등장했다. 트리니티는 기존 Hadoop 메커니즘과 다르게 MapReduce 작업을 실행하기 위한 데이터 전처리, 중간결과를 저장하지 않기 때문에 더 높은 처리량을 얻을 수 있다. 하지만 현재 트리니티에서 동작하는 서열정렬 알고리즘이 존재하지 않는다. 결과적으로 In-memory 프레임워크를 사용해 데이터를 처리하면, Hadoop의 MapReduce에서 동작하는 코드를 작성하는 것보다 더 쉽게 작성할 수 있을 뿐만 아니라, 서열 알고리즘을 사용하는 모든 알고리즘이 더 높은 처리량을 얻을 수 있을 것이다. 더 나아가생물정보학은 계속적으로 데이터의 양이 증가하기 때문에 적절한 분산처리 환경에서 효율적으로 처리하는 게 필요하다.

References

[1] Dean J. and Ghemawat S. "MapReduce: simplified data processing on large clusters", Commun. ACM 51:107-113, 2008.

[2] Schatz MC, "CloudBurst: highly sensitive read mapping with MapReduce", Bioinformatics;25:1363-9, 2009.

[3] Nguyen T. Shi W, and Ruden D, "CloudAligner: a fast and full-featured MapReduce based tool for sequence mapping", BMCResNotes, 4:171, June 2011.

[4] Langmead B., Schatz MC, and Lin J. et al., "Searching for SNPs with cloud computing", Genome Biolgy, 10:R134, Nov. 2009.

[5] Langmead B, Hansen KD, and Leek JT, "Cloud-scale RNAsequencing differential expression analysis with Myrna", Genome Biol, 11:R83, Aug. 2010.

[6] B. Shao, H. Wang, and Y. Li, "Trinity: A distributed graph engine on a memory cloud", ACM SIGMOD 978-1-4503-2037-5/13/-06, 2013.

[7] R. Li, Y. Li, K. Kristiansen, and J. Wang, "SOAP: short oligonucleotide alignment program", Bioinformatics 24 (5): 713-714 Jan. 2008.

[8] Smith A, Chung W., Hodges E., Kendall J., Hannon G., Hicks J. Xuan Z. and Zhang M., "Updates to the RMAP short-read mapping software", Bioinformatics 25(21):2841. Sep. 2009.

[9] Smith A. Xuan Z. and Zhang M, "Using quality scores and longer reads improves accuracy of Solexa read mapping", BMC Bioinformatics 9:128. Feb. 2008.

[10] Nils Homer, Barry Merriman, and Stanley F. Nelson, "BFAST: An Alignment Tool for Large Scale Genome Resequencing", PLoS ONE 4(11): e7767. doi:10.1371/ journal.pone.0007767. Nov. 2009.

[11] H. Li and R. Durbin, "Fast and accurate short read alignment with Burrows-Wheeler transform", Bioinformatics, Vol. 25, No. 14, pp. 1754-1760, July 2009.

[12] http://sqoop.apache.org/

[13] https://pig.apache.org/
[14] http://hive.apache.org/
[15] Landau, G. M. et al "Introducing efficient parallelism into approximate string matching and a new serial algorithm", In Proceedings of the Eighteenth Annual ACM Symposium on Theroy of Computing. ACM. Berkely, CA. USA. pp. 220-230, 1986
[16] Li R. Li Y. Fang X, Yang H., Wang J., and Kristiansen K., "SNP detection for massively parallel whole-genome resequencing", Genome Res 19:1124-1132, June 2009.

저자소개

이 준 수 (Jun-Su Lee)

2012년 : 경기대학교 컴퓨터과학과 졸업(학사)
2012년 ~ 현재 : 연세대학교 컴퓨터과학과 석사과정
관심분야 : 바이오인포매틱스, 데이터마이닝, 분산처리 시스템

여 윤 구 (Yun-Ku Yeu)

2009년 : 연세대학교 컴퓨터과학과 졸업(학사)
2011년 : 연세세대학교 컴퓨터과학과 졸업(석사)
2011년 ~ 현재 : 연세대학교 컴퓨터과학과 박사과정
관심분야 : 바이오인포매틱스, 데이터마이닝, 데이터베이스 시스템

노 홍 찬 (Hong-Chan Roh)

2006년 : 연세대학교 컴퓨터과학과 졸업(학사)
2008년 : 연세대학교 컴퓨터과학과 졸업(공학석사)
2008년 ~ 현재 : 연세대학교 컴퓨터과학과 박사과정
관심분야 : 데이터베이스 시스템, 플레시 SSD

윤 영 미 (Young-Mi Yoon)

1981년 : 서울대학교 자연과학대학 졸업(학사)
1983년 : 오하이오 주립대학 수학과(학사수료)
1987년 : 스탠포드대학교 컴퓨터과학과 졸업(이학석사)
1987년 5월 ~ 1993년 5월 : InteliGenetics Inc.. California USA Software Engineer
2008년 : 연세대학교 컴퓨터과학과 졸업(공학박사)
1995년 2월 ~ 현재 : 가천대학교 컴퓨터공학과 교수
관심분야 : 데이터베이스 시스템, 데이터마이닝, 바이오인포매틱스, 소셜데이터마이닝

박 상 현 (Sang-Hyun Park)

1989년 : 서울대학교 컴퓨터공학과 졸업(학사)
1991년 : 서울대학교 대학원 컴퓨터공학과(공학석사)
2001년 : UCLA 대학원 컴퓨터과학과(공학박사)
1991년 ~ 1996년 : 대우통신 연구원
2001년 ~ 2002년 : IBM T. J, Watson Research Center Post-Doctoral Fellow
2002년 ~ 2003년 : 포항공과대학교 컴퓨터공학과 조교수
2003년 ~ 2006년 : 연세대학교 컴퓨터과학과 조교수
2006년 ~ 2011년 : 연세대학교 컴퓨터과학과 부교수
20011년 ~ 현재 : 연세대학교 컴퓨터과학과 교수
관심분야 : 데이터베이스, 데이터마이닝, 바이오인포매틱스, 적응적 저장장치 시스템, 플래쉬메모리 인덱스, SSD