하둡을 이용한 BLAST의 병렬 처리 및 성능 분석

최동훈, 엄정호, 윤화묵, 최윤수, 이민호, 이원구, 송사광, 정한민

한국과학기술정보연구원 소프트웨어연구실
e-mail:choid@kisti.re.kr

Parallel Processing of BLAST Using Hadoop and Its Performance Evaluation

Hoon Choi, Jungho Um, Hwa-mook Yoon, Yun-Soo Choi, Minho-Lee,
Won-Goo Lee, Sa-Kwang Song, Hanmin Jung

Dept. of Software Research, KISTI

요 약

차세대 시퀀싱 장비는 기존의 컴퓨팅 방법으로 처리할 수 없을 만큼 많은 양의 시퀀스 데이터를 생성하고 있다. 본 논문에서는 차세대 시퀀스 데이터의 정렬을 위해 널리 사용되고 있는 BLAST의 병렬 처리 방법을 하둡을 사용하여 제시하며, 이의 성능 개선 효과를 분석한다.

1. 서론

차세대 시퀀싱 장비는 기존의 컴퓨팅 방법으로 처리할 수 없을 만큼 많은 양의 시퀀스 데이터를 생성하고 있다. 이러한 데이터의 양적 특징은 소프트웨어의 발전에도 불구하고, 시퀀스의 정렬(alginment)을 위해 더 많은 용량의 컴퓨팅 자원을 요구하고 있다. 최근 들어 많은 사람에 의해 사용의 편이성과 비용의 효율성 등이 이점으로 부각되고 있는 클라우드 컴퓨팅 기술은 이러한 문제를 해결할 수 있는 대안으로 떠오르고 있다.

BLAST는 대규모 시퀀스 데이터의 정렬을 위해 바이오 인포매틱스 연구자가 일반적으로 널리 사용하는 프로그램이다. 연구자는 BLAST를 통해 시퀀스 데이터 간의 유사도를 계산하고, 그 결과를 파싱하여 유사도가 일정 수준을 넘는 것만을 선택하여, 이후의 시퀀스 분석 절차를 수행한다. 클라우드 컴퓨팅을 통해 시퀀스 데이터의 정렬을 고속 처리하기 위한 시도로 CloudBLAST[1], CloudBurst[2] 등이 있다. 시퀀스 데이터의 정렬은 데이터 분할에 의한 병렬 처리가 가능하므로, 하둡(Hadoop)을 사용하여 병렬 처리할 수 있다[3]. 이들은 모두 하둡을 기반으로 BLAST를 병렬 처리하였으며, CloudBurst는 아마존 웹 서비스로 제공되고 있을 정도로 널리 알려져 있다.

본 논문에서는 하둡을 이용하여 차세대 시퀀스 데이터의 정렬을 위해 가장 많이 사용되고 있는 BLAST의 병렬 처리 방법과 이의 성능 개선 효과를 분석하고자 한다. 이에 대한 성능 평가를 위해 차세대 시퀀싱 장비에 의해 생성된 고추의 시퀀스 데이터의 샘플을 사용하며, 이 데이터 샘플은 한국생명공학연구원과의 공동 연구[4]를 통해 입수되었다.

2. 하둡 기반의 BLAST 병렬 처리

2.1 BLAST에 의한 시퀀스 정렬 절차

BLAST에 의한 시퀀스 데이터의 정렬은 다음과 같은 절차를 거친다.

① BLAST 실행하여 시퀀스 매치에 의한 유사도를 계산한다.

② BLAST 결과에 파싱 조건을 적용하여 선택된 것을 시퀀스 ID 쌍의 형태로 저장한다.

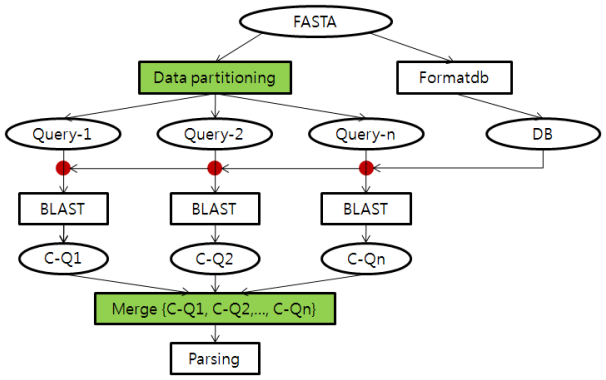
BLAST를 이용한 시퀀스 매치 결과는 시퀀스 ID1, 시퀀스 ID2, 일치율, 일치되는 염기의 수, 쿼리(fasta file)의 일치되는 시퀀스의 시작과 끝, DB(formatDB)의 일치되는 시퀀스의 시작과 끝을 포함한다. 일치율과 일치되는 염기의 수를 비롯한 위의 정보를 바탕으로 이후 분석에 필요한 시퀀스 쌍을 골라낼 수 있도록 조건문을 설정하게 되는데, 이것이 파싱 조건이 된다. BLAST의 시퀀스 매치 결과 중 시퀀스 ID 쌍을 제외한 다른 정보들은 더 이상 사용되지 않기 때문에 파싱 결과에서 제외한다.

2.2 하둡 스트리밍을 이용한 병렬 처리

하둡 프로그램은 Map과 Reduce 함수로 구성되며, 하둡 실행 시스템은 이들을 사용자가 지정한 만큼의 타스크를 생성하여 병렬 처리를 수행한다. 하둡을 이용하기 위해서는 BLAST 알고리즘에 Map과 Reduce 함수를 정의

하여 프로그램을 새로 짜거나 하둡 스트리밍(Hadoop streaming)을 사용하여 처리한다. 기존의 프로그램을 Map과 Reduce를 사용하여 다시 개발하는 것은 많은 시간과 인력을 필요로 하며, 프로그램의 구조가 하둡 방식으로 바꾸기 적합하지 않을 수도 있다. 이러한 이유로 대다수의 경우, 기존의 프로그램을 mapper와 reducer로 이용하는 하둡 스트리밍 방식을 사용하여 병렬 처리를 수행한다. 실제로 하둡을 이용하는 대다수의 프로젝트는 Map과 Reduce에 의한 재개발이 아닌 하둡 스트리밍을 사용하고 있다. 특히, BLAST와 같이 복잡도가 높은 프로그램은 Map과 Reduce를 사용하여 새로 구현하는 것보다 하둡 스트리밍을 사용하는 것이 유리하다.

그림1은 하둡 스트리밍을 이용한 BLAST의 병렬 처리 방식을 보여주고 있다.



(그림 1) 하둡을 이용한 BLAST의 병렬 처리

3. 성능 분석

3.1 실험 환경

4대의 서버를 이용하여 하둡 클러스터를 구성하였다. 한 대의 서버를 하둡의 마스터 노드로, 나머지 3대의 서버를 슬레이브 노드로 사용하였다. 각 서버는 네트워크로 연결되어 있으며 서버의 사양은 <표1>과 같다.

<표1> 서버 사양

CPU	Intel(R) Xeon(R) CPU E5540 @ 2.53 GHz L1 cache: 256KB, L2 cache: 1MB 8 core (2 cpu X 4 core)
메모리	16 GB (4 GB X 4), DIMM 1333 MHz
NIC	Intel Gigabit Network Card
HDD	6 TB (1TB X 6) (실제 사용 가능 용량 4.5 TB)

생명공학연구원에서 제공한 염기서열의 시퀀스 파일 s_1_1_sequence.seq와 xau의 데이터를 이용하여 실험을 수행해 보았으며 데이터의 요약은 <표2>와 같다.

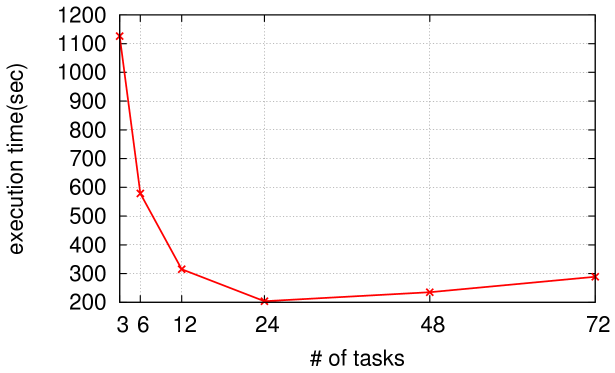
<표2> 데이터 요약

	xau 파일	s_1_1_sequence.seq
원본 시퀀스 수	443,048	20,443,048
원본 데이터 용량	46 MB	2.1 GB
Filtering 후 시퀀스 수	1,195,105	500,167,265
Filtering 후 데이터 용량	72 MB	28 GB

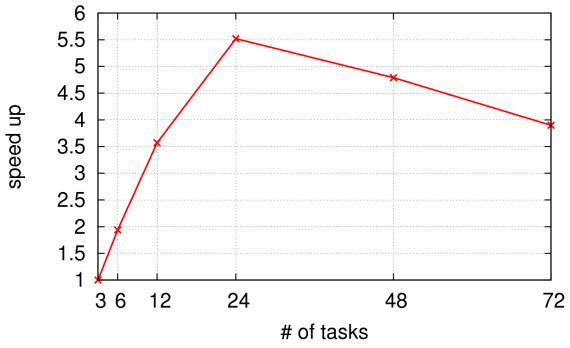
3.2 성능 분석 결과

- 하둡 타스크의 수에 대한 성능 분석

타스크 수를 증가시키면서 같은 양의 데이터를 처리하는데 걸리는 시간을 분석하면 그림2와 같다. 이것은 타스크 수가 한 서버당 코어 수가 8개이므로 8개가 적합하다는 것을 보여준다.



(그림 2) 하둡을 이용한 BLAST 실행 시간



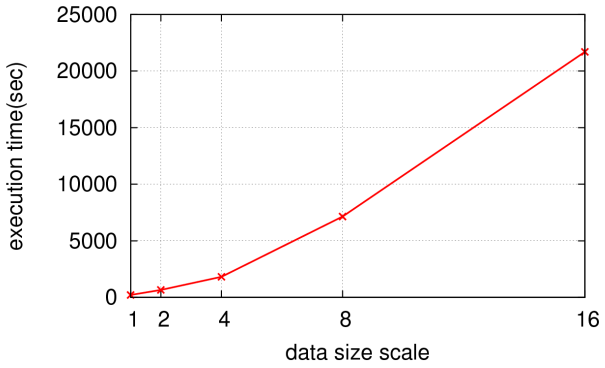
(그림 3) 확장성(scalability)

그림 4는 하둡을 이용한 BLAST 병렬 처리의 확장성(scalability)을 보여준다. 실험의 편의를 위해 s_1_1_sequence.seq 파일의 1/64 크기의 데이터를 만들어서 BLAST 프로그램을 수행하였다. 이는 s_1_1_sequence.seq 데이터가 20,443,048 개의 시퀀스로 이루어져 있으므로 1/64인 319,422 개의 시퀀스를 포함하는 파일을 생성한

것이다. 하나의 데이터를 처리하는데 1개의 태스크를 사용할 경우 3,182 초가 걸릴 작업을 3개의 태스크를 사용하면 1126초, 24개의 태스크를 사용하면 204초가 걸렸다. 한편, 태스크의 수가 24개를 넘어가는 순간부터 수행시간이 오히려 더 이상 향상되지 않음을 볼 수 있었다. 타스크의 수가 24개까지는 어느 정도 확장성을 만족시킴을 알 수 있다. 이는 4대의 서버에서 최대 성능 효율을 낼 수 있는 타스크의 수를 의미한다. 만약 서버 노드의 수를 더 늘리게 되면 그만큼 타스크의 수도 비례하여 증가하게 되므로 성능 또한 거의 비례하게 증가함을 쉽게 예상할 수 있다.

- 데이터 크기 대비 성능 비교

데이터 크기가 증가함에 따라 프로그램 수행하는 시간이 얼마나 증가하는지를 살펴보기 위해서 20,443,048 개의 시퀀스로 이루어져 있는 s_1_1_sequence.seq 파일을 1/64 등분하여 319,422개의 시퀀스로 이뤄진 파일을 scale=1로 가정하여, 638,844 개의 시퀀스로 이뤄진 파일(scale=2), 1,277,688 개의 시퀀스로 이뤄진 파일(scale=4), 2,555,376 개의 시퀀스로 이뤄진 파일(scale=8), 5,110,762개의 시퀀스로 이뤄진 파일(scale=16)을 생성하였다. 각 파일을 BLAST로 돌렸을 때 걸린 시간을 측정한 결과 그림4와 같다. 데이터의 크기가 늘어남에 따라 수행시간이 다소 비례적으로 증가함을 볼 수 있다.



(그림 4) 데이터 크기 증가에 따른 실행 시간

4. 결론

본 논문에서는 하둡을 이용하여 시퀀스 데이터의 정렬에 널리 사용되고 있는 BLAST의 병렬 처리 방법과 이의 성능 개선 효과를 분석하였다. 4대의 8-코어 리눅스 머신 위에 하둡을 설치하여 BLAST의 병렬 처리 방법을 개발하였으며, 고추 염기 서열 데이터를 사용하여 성능 분석을 통해 그 효과를 검증하였고, 다음과 같이 향후 연구의 방향성을 살펴 볼 수 있었다.

하둡 클러스터에서 효율적인 병렬 연산을 하기 위해서는, 각 작업 노드들의 성능이 균등하다는 가정 하에, 그 작업량을 균등 분배할 필요가 있다. 그러나 이질적인 컴퓨팅 환경에서 작업을 균등 분배하기 위해서는 별도의 계산 과정이 필요하며, 이를 위한 최적화 문제도 해결해야 한다. 또한 하둡은 내고장성을 위해 Map과 Reduce의 각 단계마다 결과를 로컬 디스크에 기록하거나 또는 3개 이상의 사본을 유지하는 HDFS에 결과를 쓰도록 되어 있다. 이러한 방식은 많은 디스크 IO를 야기하며 이에 따른 성능 저하가 발생할 수 있다. 이를 해결하기 위해 디스크 IO 절감 방법을 연구하는 것은 하둡의 효율을 제고에 필수적이다.

참고문헌

[1] A. Matsunaga, M. Tsugawa and J. Fortes, “CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications,” Fourth IEEE International Conference on eScience, 2008

[2] M. Schatz, “CloudBurst: highly sensitive read mapping with MapReduce,” Bioinformatics 25(11): 1363-1369, 2009

[3] Hadoop. http://hadoop.apache.org/

[4] 허철구, 클라우드 컴퓨팅을 이용한 유전자 서열 데이터의 contig assembly에 관한 연구, 한국생명공학연구원, 2010