생물정보학에서 사용자 그룹핑 기법을 이용한 서열 정렬 방법

김민준°, 김재훈
아주대학교 정보 통신 전문 대학원
xwind@dmc.ajou.ac.kr jaikim@madang.ajou.ac.kr

Sequence Alignment Scheme Using User Grouping for Bioinformatics

Min Jun Kim°, Jai-Hoon Kim
Graduate School of Information and Communication Ajou University

요 약

생물정보학 관련 프로그램들은 대부분 인터넷을 통해서 많은 생물학자, 생화학자 들에게 생물 정보 검색 및 처리 서비스를 제공한다. 이런 서비스를 제공하는 많은 프로그램들은 데이터베이스로부터 유전자 등의 데이터를 검색하고 처리한다. 이때 각각 클라이언트의 요청마다 매번 데이터베이스의 검색을 수행한다면 많은 시간이 걸리게 된다. 또한 서버에 과부하를 초래하여 응답시간이 길어 질 수 있다. 본 논문에서는 서버의 과부하와 응답시간을 줄이기 위해 사용자의 요청을 그룹화하여 일정 시간 간격으로 일괄 처리하는 방법을 제안한다.

1. 서론

21세기 초에 인간 유전자 프로젝트의 성공적인 수행은 모든 생명과학 분야의 급속한 발전을 야기시켰으며, 이러한 인간유전체 지도의 완성으로 전개되는 유전자이후시대(post Genom)에는 인간의 모든 유전자와 유전자의 발현으로 생성되는 단백질들의 구조와 기능에 관한 연구가 활발히 수행될 것이다. 이러한 요구조건에 맞게 유전자를 비교 분석 연구하기 위한 많은 소프트웨어가 개발되었다. FastA, Blast, ClustalW등의 패턴 매치 프로그램과 J-NET이나 J-PRED등의 구조예측 프로그램등은 좋은 예이다. 이러한 프로그램들은 gene-bank등의 데이터베이스 기반으로 작동으로 하는데 이 때문에 디스크 액세스 시간에 많은 시간을 할당해야만 한다. 경우에 따라 계산 시간보다 훨씬 많은 디스크 액세스 시간 때문에 많은 경우 성능이 저하 될 수 있다.

본 논문에서 사용자 요청에 대한 그룹핑을 통해서 디스크 액세스를 줄이고 소프트웨어들의 성능을 향상 시킬 수 있는 방법을 제안한다. 디스크 액세스가 많은 VOD시스템에서 많은 사용자 요청을 효율적으로 처리하기 위한 방법으로 PDP 알고리즘을 사용한다. 이 방식의 VOD서버는 AT&T에서 현재 구현되어 있다[1]. PDP는 활동 그룹이라는 사용자 요청의 집합을 만들고 각 사용자 요청에 대해서 이미 플레이 된 영역을 캐쉬하고 다음 읽을 영 역을 일정 시간 동안 캐쉬하게 된다[1].본 논문에서는 VOD에서 사용하는 사용자 요청 그룹화와 비슷한 방식을 사용하여 생물정보학관련 프로그램들의 성능을 향상 시키고자 한다.

2. 일반적인 서열 정렬

Fast A나 Blast등의 프로그램들은 웹을 통해 서비스를 하며 사용자가 서버에 접속하여 비교할 단백질 서열을 서버에 보내게 된다. 서버는 데이터베이스에서 서열을 읽어들여 사용자가 요청한 서열과 비교하게 된다.

2.1 일반적인 구조

생물정보학에서 사용하는 프로그램들은 데이터베이스 기반으로 작동을 한다. 즉, 매번 사용자의 요청 마다 데이터베이스에 접근하여 데이터를 읽은 후 사용자의 요구에 응답을 해야 한다. 예를 들어 FastA의 경우 사용자의 요구가 있을 시 사용자는 비교하고자 하는 단백질 서열을 서버에 보낸다. 서버는 사용자의 요청이 있을 때, 데이터베이스를 블록 단위로 읽어서 각 블록에 들어 있는 서열과 사용자가 보낸 서열을 비교한다. 그리고 비교한 결과를 사용자에게 보낸다.

2.2 수학적인 분석

데이터베이스에서 한 블록을 읽어 들일 때 소요되는 디스크 액세스 시간을 C_{io} 라 정의한다. 그리고 서열 비교 대상이 되는 전체 데이터베이스의 블록의 개수를 N_b 라 정의한다.

또한, 블록 안에 포함되어 있는 단백질 서열의 개수는

$N_{s/b}$라 정의한다. 사용자의 서열과 데이터베이스에서 읽어 들인 하나의 단백질 서열과 사용자가 요청한 단백질 서열간의 비교 시간 즉, 프로세싱 시간을 C_{cpu}로 정의한다.

데이터베이스는 하나의 단백질 서열을 받을 때 마다 데이터베이스의 모든 내용을 메모리로 가져와야 한다. 이때 걸리는 시간은 데이터베이스에서 한 블록을 읽어 들이는 시간과 전체 데이터베이스의 블록의 개수의 곱과 같다. 한 블록을 읽어 들일 때의 시간은 모두 같다고 가정한다. 그러면 디스크 액세스 시간을 아래와 같이 나타낼 수 있다.

$$C_{io} \times N_b$$

위의 수식은 데이터베이스 검색을 위한 디스크 액세스의 시간이다. 그리고 각 서열간의 비교 시간은 사용자가 요청한 하나의 서열을 디스크에서 읽은 비교 대상이 되는 서열과 비교하는 시간이다. 모든 서열의 개수는 블록 당 서열의 수와 전체 블록을 개수로 나타낼 수 있다. 즉, 전체 비교 시간은 다음과 같다.

$$N_b \times N_{s/b} \times C_{cpu}$$

즉, 한 사용자가 서버에 접속하여 하나의 단백질 서열을 비교하는데 걸리는 평균 시간은 다음과 같다.

$$C_{avg}^{o} = C_{io} \times N_b + N_b \times N_{s/b} \times C_{cpu} \quad (1)$$

결국, 식 (1)은 일반적인 생명 정보학 프로그램들이 사용자 요청을 처리하는데 드는 시스템 비용(cpu시간과 디스크 액세스 시간의 합)이 된다.

또한 이 값을 다른 사용자와 시스템 자원 사용시 충돌이 일어나지 않는 다고 가정 할 때, 한 사용자가 처리를 요청하고 끝날 때까지 기다리는 시간이 된다.

3. Delayed Request Group Algorithm

3.1 Delayed Request Group Algorithm의 구조

Delayed Request Group Algorithm(이하 DRGA)은 매번 디스크 액세스를 하지 않고 일정한 시간 동안 도착하는 사용자의 요청을 모아서 주기적으로 처리한다. 주기적으로 데이터베이스를 한번만 읽어 들이고 데이터베이스에서 읽어 들인 서열들은 모아진 사용자 요청

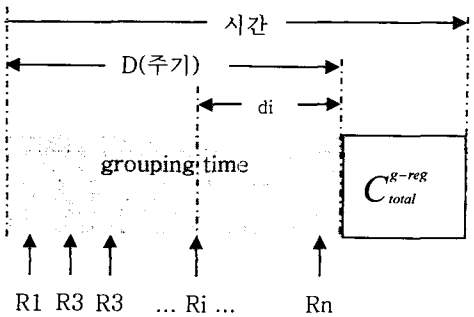

그림 1 그룹핑 처리

서열과 각각 비교를 한다. 이럴 경우 데이터베이스를 모든 요청 때마다 읽어 들일 필요가 없고 한 주기에 한 번만 읽으므로 디스크 액세스 횟수를 현저히 줄일 수 있다.

그림 1에서 R1, R2, R3, R4는 사용자 요청이 들어온 시각을 말한다. 위의 그림은 DRGA를 사용할 때 사용자 요청들이 기다리게 되는 것을 보여준다. 한 주기(D) 안에서 모든 요청들은 한번에 처리 된다. 임의의 사용자 요청(Ri)들은 주기(D)가 끝날 때까지 di시간동안 기다리게 된다.

3.2 DRGA의 수학적 분석

일정 주기(D)동안 사용자의 요청을 한번에 처리할 때 소요되는 시스템 비용(cpu시간과 디스크 액세스 시간의 합)에는 사용자가 처리를 기다리는 시간(di)이 포함되지 않는다. 이를 수식으로 표현하면 아래와 같이 나타낼 수 있다.

$$C_{total}^{g-reg} = C_{io} \times N_b + N_b \times N_{s/b} \times \sum_{i=1}^{D*N} C_{cpu}$$

일정 주기(D) 동안 사용자 요청 개수는(사용자 요청의 도착율) 단위 시간 동안 평균 N번이라 할 때 D × N이 된다. 디스크 액세스는 모든 데이터를 1회 읽으면 되고 ($C_{io} \times N_b$), 서열 비교시간($N_b \times N_{s/b} \times \sum_{i=1}^{D*N} C_{cpu}$)이 소요된다. DRGA기법을 사용할 경우 사용자 요청 당 평균 시스템 비용은 다음과 같다.

$$C_{avg}^{g-reg} = \frac{C_{total}^{g-reg}}{D*N} = \frac{C_{io} * N_b}{D*N} + N_b \times N_{s/b} \times C_{cpu} \quad (2)$$

만일, 시스템 비용대신 사용자가 처리를 요청하고 처리가 끝날 때까지 기다리는 평균 시간을 계산한다면 다음과 같다.

$$C_{avg}^{g-reg,\prime} = \frac{C_{io} N_b + N_b N_{s/b} C_{cpu} N D}{2} + \frac{D}{2} \quad (2)'$$

3.3 성능 비교

그림 3은 성능 비교를 위해서 식 (1)과 식 (2)에 값을 대입하여 본 결과이다. C_{io}는 데이터베이스 액세스 시간으로서 디스크 액세스가 일어나므로 10ms로 가정하였다. 단백질 서열을 디스크 한 블록에 저장할 수 있다고 가정하여 블록 당 서열의 개수인 $N_{s/b}$는 1로,

데이터베이스 블록의 개수인 N_b는 10,000로 가정하며 서열간에 비교를 하는 CPU시간인 C_{cpu}는 10ms으로 가정하였다. DRGA를 사용하지 않고 사용자의 요청 시마다 서열 데이터베이스 액세스와 서열 정렬을 수행하는 경우와 주기마다 사용자 요청을 일괄 처리하는 DRGA의 시스템 비용을 비교 하였다.

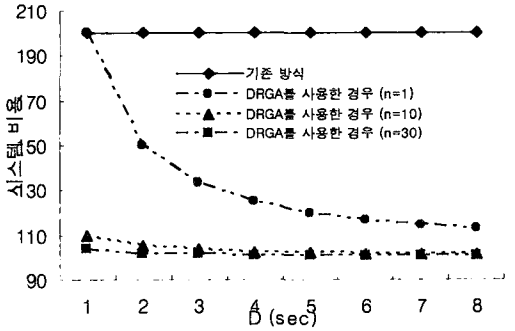

그림2 DRGA의 성능 비교

그림 2에서 보는 바와 같이 DRGA를 사용한 경우 주기(D)에 값에 따라 사용자 요청을 처리하는 평균 시스템 비용이 줄어든 것을 볼 수 있다. 기존 방식은 일정한 시스템 비용이 소요되지만 DGRA를 사용할 경우 사용자 도착율(N)과 주기(D)가 클수록 처리비용이 감소하는 것을 볼 수 있다.

3.4 DRGA의 효율과 데이터베이스의 크기

기존의 기법의 경우 단위시간당 N개의 사용자 요청이 들어올 때 단위 시간당 시스템 비용은 식 (1) × N으로 표현할 수 있다. 시스템 이용율은 1을 넘을 수 없으므로 다음 조건을 만족해야 한다. (이때, 디스크 액세스와 CPU를 순차적으로 처리한다고 가정하였다.)

$$(C_{io} \times N_b + N_b \times N_{s/b} \times C_{cpu}) \times N \leq 1$$

이를 N에 대해서 정리하면 다음과 같다.

$$N \leq \frac{1}{C_{io} N_b + C_{cpu} N_b} \quad (3)$$

같은 방법으로 DRGA의 처리 비용을 나타내는 식(2) × N ≤ 1 를 정리하면 다음과 같다.

$$N \leq \frac{1}{C_{cpu} N_b} - \frac{C_{io}}{D\, C_{cpu}} \quad (4)$$

식(3)과 식(4)를 이용하여 데이터베이스의 크기(N_b)에 따른 처리 가능한 최대 사용자 도착율을 비교해 보면 그림3와 같다.

그림 3에서 보는 바와 같이 DRGA는 기존의 방법보다 좋은 처리율을 갖는다. 또한 주기(D)를 길게하여 처리 율을 높일 수 있다.

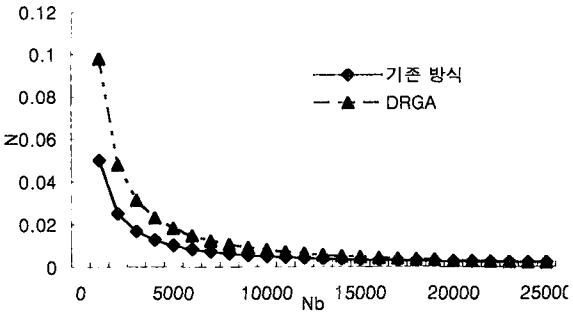

그림3 효율비교

그림 3은 D를 500으로 가정하였을 때 기존의 방법과 DGRA를 사용하였을 때의 처리율을 보여준다.

4. 결론

단백질 데이터베이스는 매우 빠르게 증가 하고 있다. 단백질 데이타의 폭발적인 증가는 컴퓨터의 발전 속도를 능가할 정도이며 또한 이를 분석하기 위해서도 데이터베이스의 잦은 액세스는 생물 정보학 관련 문제들을 처리함에 있어 과부하로 작용할 것이다. 본 논문에서 제안한 DRGA을 서버가 사용자 요청을 처리하는 비용을 줄일 수 있을 뿐 아니라 데이터의 폭박적인 증가에 대한 대처가 가능하다.

데이터베이스의 액세스를 줄이고 서버의 과부하를 없엘 수 있을 뿐 아니라 이로 인해 더 많은 사용자들에게 서비스를 제공할 수 있음은 자명한 일이다.

참고문헌

[1] Ozden, B. Rastogi, R. Silberschatz, A. Martin, C.. " demand Paging for Video-on-demand Servers. Multimedia Computing and Systems 1995", Proceedings of the International Conference on , 1995

[2] 김진, 최홍식, 류웅진 " 분자 생물학과 알고리즘", 정보과학회지 18(8) 2000.8

[3] 김양석, " 게놈 프로젝트를 위한 생물 정보학" 정보과학회지 18(8) 2000.8

[4] DataGrid, " Grid-Aware Bicmedical Applicatinos for DataGrid Testbed Assessment", DataGrid-100d10.2-0109-2-0, 2002.03.04

[5] Stephen F Altschul, Thomas L. Madden, Alejandro A. Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, David J. Lipman, " Gapped Blast and PSI-Blast: a new generation of protein database search programs", Oxford University Press, 15(17), 1997