韓國 컴퓨터情報學會 論文誌
第11卷 第6號, 2006. 12

2006-11-6-1-8

클러스터 환경에서의 MPI 기반 병렬 서열 유사성 검색에 관한 연구

홍창범*, 차정호*, 이성훈**, 신승우**, 박근준***, 박근용****

Study on MPI-based parallel sequence similarity search in the LINUX cluster

Chang-Bum Hong*, Jeoung-Ho Cha*, Sung-Hoon Lee**, Seung-Woo Shin**,
Keun-Joon Park***, Keun-Young Park****

요 약

생물정보학 연구 있어서 아미노산이나 염기서열에 대한 유사성이나 상동성을 찾아내는 작업은 유전자의 기능에 대한 예측이나 단백질 구조를 예측하는 연구의 기반이 된다. 이러한 서열 데이터는 컴퓨터의 도입으로 매우 빠르게 증가하고 있다. 이러한 시점에서 서열에 대한 검색 속도는 매우 중요한 요소이기 때문에 대량의 서열정보를 다루기 위해서는 SMP(Sysmmetric Multi-Processors) 컴퓨터나 클러스터를 이용하고 있다. 본 논문에서는 서열 검색에 사용되는 BLAST(Basic Local Alignment Search Tool)의 속도향상을 위한 방법으로 클러스터 환경에서 병렬화 하는 nBLAST 알고리즘의 병렬화에 대해 제안한다. nBLAST는 기존의 BLAST 소스코드에 대한 수정 없이 병렬라이브러리인 MPI(Message Passing Interface)를 이용하여 질의를 분할하여 병렬화 하기 때문에 환경설정 등의 복잡한 과정을 거치지 않고 손쉽게 BLAST에 알고리즘에 대한 병렬화를 할 수 있다. 또한, 실험을 통하여 28대의 리눅스 클러스터에서 nBLAST를 수행하여 노드 수의 증가에 따른 성능 향상을 확인하였다.

Abstract

In the field of the bioinformatics, it plays an important role in predicting functional information or structure information to search similar sequence in biological DB. Biological sequences have been increased dramatically since Human Genome Project. At this point, because the searching speed for the similar sequence is highly regarded as the important factor for predicting function or structure, the SMP(Sysmmetric Multi-Processors) computer or cluster is being used in order to improve the performance of searching time. As the method to improve the searching time of

• 제1저자 : 홍창범
• 접수일 : 2006.11.06, 심사일 : 2006.11.18, 심사완료일 : 2006. 12.20
* 질병관리본부 국립보건연구원 기술연구원
** 질병관리본부 국립보건연구원 선임연구원
*** 질병관리본부 국립보건연구원 책임연구원
* 질병관리본부 국립보건연구원 기술연구원
** 질병관리본부 국립보건연구원 선임연구원
**** 질병관리본부 국립보건연구원 보건연구관
※본 논문은 2006년도 질병관리본부 조사연구사업 "바이오과학정보관련 DB 산출물의 표준을 위한 기준 설정 및 시스템 성능 향상 방안 연구"에 의해 연구되었음.

BLAST(Basic Local Alighment Search Tool) being used for the similarity sequence search, We suggest the nBLAST algorithm performing on the cluster environment in this paper. As the nBLAST uses the MPI(Message Passing Interface), the parallel library without modifying the existing BLAST source code, to distribute the query to each node and make it performed in parallel, it is possible to easily make BLAST parallel without complicated procedures such as the configuration. In addition, with the experiment performing the nBLAST in the 28 nodes of LINUX cluster, the enhanced performance according to the increase in the number of the nodes has been confirmed.

▸ Keyword : 생물정보학(Bioinformatics), MPI(Message Passing Interface), 서열검색(Sequence Search)

Ⅰ. 서론

생물정보학 연구에 있어서 아미노산이나 염기서열에 대한 서열의 유사성 검색으로 상동성을 찾아내는 작업은 유전자의 기능을 예측하거나 단백질의 2차 구조의 예측 및 분자 모델링 디자인 연구에 기반이 된다. 이러한 서열 검색 작업에서 동적프로그래밍 알고리즘은 최선의 정렬을 찾기 위한 모든 가능성을 결정하기 때문에 항상 최적의 결과를 얻을 수 있다. 그러나 많은 결정에 대한 순서들이 생성되기 때문에 이를 구현하기 위해서는 많은 메모리와 시간을 필요로 한다[1]. 따라서 휴리스틱 접근 알고리즘을 사용하여 결과의 정확성 대신 속도를 얻는 BLAST(Basic Local Alignment Search Tool) 알고리즘이 서열 검색에 널리 사용되고 있다[2]. BLAST 알고리즘의 특징은 첫째, 알려지지 않은 서열에서 유용한 정보를 확인하는데 강력한 도구라는 점과 둘째, 빠르게 작동한다는 점 셋째, 통계적 입장과 소프트웨어 개발 관점 두 가지에서 정확하다는 점과 마지막으로 BLAST는 유연하고 많은 서열 분석 시나리오를 적용할 수 있다는 것이다[3,4]. BLAST 알고리즘을 구현한 프로그램으로는 NCBI BLAST와 WU-BLAST의 두 가지 버전이 널리 사용되고 있다. NCBI BLAST는 미국생명공학 정보센터(National Center for Biotechnology Information)에서 다중 서열의 프로파일 비교에 초점을 두고 만들어졌으며, WU-BLAST는 Washington 대학의 Dr. Warren Gish가 NCBI BLAST 버전 1.4를 기본으로 탐색과 갭 처리에서 NCBI BLAST와는 다른 방식을 적용하였다[5]. 본 논문에서 NCBI BLAST를 중심으로 분석하였다.

생물학에서의 서열에 대한 시퀀싱 자동화와 컴퓨터 프로그래밍의 도입으로 서열 데이터는 매우 크고 그 성장 속도 또한 빠르게 이루어지고 있다. 이러한 시점에서 서열 검색에 대한 속도는 매우 중요한 요소로 대두되게 되었다. 현재 생물정보학 분야 등 복잡하고 어려운 문제를 다루는 곳에서 문제를 빠르게 처리하기 위한 방법으로 SMP(Sysmmetric Multi-Processors) 컴퓨터나 클러스터 컴퓨터를 활용하고 있다. 또한 인터넷과 네트워크의 비약적인 발전으로 전 세계적으로 지역의 제한 없이 모든 컴퓨팅 자원을 연결하여 사용하는 그리드 컴퓨팅을 이용하는 방법이 현재 활발히 연구되고 있다[6].

지금까지 BLAST의 속도 향상을 위한 방법으로는 하드웨어 가속 기능을 사용하거나 서열 데이터베이스를 분할하는 방법에 대한 연구가 진행되고 있다. 이러한 방법은 별도의 하드웨어의 구입이나, 여러 가지 환경 설정 등의 복잡한 과정을 거쳐 실행되기 때문에 연구자들이 사용하기에 어려움이 따른다는 단점이 있다.

본 논문에서는 클러스터 환경에서 MPI(Message Passing Library)를 사용하여 BLAST 질의 서열을 분할하여 병렬 수행하는 nBLAST를 제안한다. nBLAST는 기존의 BLAST에 대해서 수정 작업 없이 바로 적용이 가능하기 때문에 BLAST가 설치된 클러스터에서 별도의 작업을 거치지 않고 BLAST의 병렬화를 통해 BLAST의 속도 향상을 증가시킨다는 장점이 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 BLAST 알고리즘과 기존 병렬화 방법, 병렬프로그래밍에 대해 알아본다. 3장에서는 본 논문에서 제안하는 리눅스 기반의 클러스터 시스템에서 질의 분할 방법을 통한 병렬화 알고리즘을 제안한다. 4장에서는 실제 28노드로 구성된 kgene 클러스터에서의 구현에 대하여 알아본다. 5장에서는 kgene 클러스터 시스템에서 실제 성능을 측정한 결과를 제시하고 6장에서는 결론을 맺는다.

II. 관련 연구

본 장에서는 BLAST 알고리즘과 BLAST 병렬화 연구에 대해 기술한다.

2.1 BLAST 알고리즘

BLAST는 공개된 서열 데이터베이스 내에서 분석하고자 하는 서열과 비교를 통해 유사성을 지닌 아미노산이나 염기 서열을 찾고 그에 대한 기능적 특징을 알아내는데 목적이 있다.

NCBI BLAST는 blastall이라는 하나의 실행파일을 통해 blastn, blastp, blastx, tblastn, tblastx의 서로 다른 다섯 가지 프로그램을 하나의 인터페이스를 이용하여 수행할 수 있다. 이러한 NCBI BLAST 프로그램은 (표 1)과 같이 검색하려는 데이터베이스와 질의 서열의 성질에 따라 나누어진다.

표 1. BLAST 프로그램
Table. 1 Traditional BLAST programs

Program	Database	Query
blastn	Nucleotide	Nucleotide
blastp	Protein	Protein
blastx	Protein	Nucleotide translated into protein
tblastn	Nucleotide translated into protein	Protein
tblastx	Nucleotide translated into protein	Nucleotide translated into protein

BLAST 알고리즘은 (그림 1)과 같이 검색을 원하는 질의 서열로부터 3개의 아미노산이나 11개의 염기로 이루어진 짧은 서열(words)을 만든다. 이때 만들어지는 짧은 서열은 식(1)을 통해 구할 수 있다. 예를 들어 500-base의 아미노산의 경우 총 498개의 짧은 서열을 만들 수 있다.

Maximum words = L - w + 1
(L = 질의 서열의 총 길이, w = 3(아미노산), 11(염기)) ····· 식(1)

생성된 짧은 서열은 두 아미노산의 유사성을 나타내는 매트릭스인 BLOSUM(Blocks Amino Acid Substitution Matrices)62를 사용하여 경계 값(T) 이상의 점수를 기록하는 모든 목록을 구성한다. BLOSUM62는 한 아미노산이 다른 아미노산으로 바뀔 가능성 즉 두 아미노산의 유사성을 나타낸다. BLOSUM62 점수행렬에서 양수는 두 아미노산이 서로 잘 바뀔 수 있는 경우를 의미하며, 음수는 두 아미노산이 서로 잘 바뀌지 않는 경우를 의한다. 0은 특별한 의미 없이 두 아미노산이 우연히 바뀔 수 있는 경우를 의미한다. 두 서열을 비교해서 점수가 높으면 친족관계(homology)가 더 있다고 볼 수 있다[7].

경계 값 이상의 짧은 서열 목록과 서열 데이터베이스에서 일치하는 서열을 찾아 서열 데이터베이스에서 양쪽 방향으로 갭이 없는 로컬 정렬 방식으로 확장하게 된다. 확장을 마친 후 서열 데이터베이스의 서열 중 일정 값 이상의 HSP (High-scoring Segment Pair)를 가진 서열들을 추출한다. 이때 중복되지 않는 각각의 HSP들은 통계적인 테스트를 거쳐 연결되어 최종 결과를 생성한다[3,4].

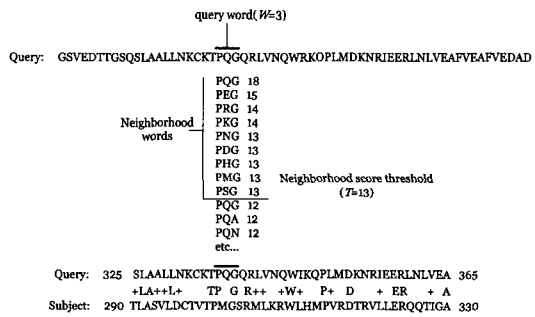

그림 1. BLAST 검색 알고리즘
Fig. 1 BLAST search algorithm

2.2 BLAST의 속도 개선

1990년대 인간 유전체 프로젝트의 시작으로 유전자 서열 데이터베이스의 크기가 급격히 증가 하면서 BLAST의 검색시간 단축은 생물정보학에서 중요한 요소로 자리 잡게 되었다. 현재 NCBI BLAST는 SMP 호스트 기반의 병렬화가 주로 이루어져 SMP 호스트의 쓰레드에 기반한 병렬화가 이루어져 있다. 그러나 프로세서의 개수가 제한된 SMP의 특성으로 확장성에 제한을 가지는 문제가 있다[8].

2.3 BLAST의 병렬화

본 절에서는 BLAST에 대한 속도향상을 위한 병렬화 방법을 살펴본다.

2.3.1 하드웨어 병렬화

하드웨어 가속 기능을 통한 BLAST의 속도 향상은 R.K. Singh[9]에 의해 처음 보고 되었으며, TimeLogic 사의 DeCyper 악셀레이터는 BLAST의 최적화를 위해서 BLAST 알고리즘을 FPGA(Field Programmable Gate Array)에 이식시켜 이를 컴퓨터에 장착하여 사용한다. DeCypher 엑셀레이터는 FPGA 보드에 최적화한 Tera-BLAST를 수행하여 BLAST 알고리즘을 실행한다[10]. 이러한 하드웨어 기반의 BLAST의 속도 향상은 많은 양의 BLAST 검색 시 유용하게 사용되며 클러스터 보다 저렴한 비용으로 구축할 수 있다는 장점이 있지만, BLAST 작업에만 특화되어 범용성이 떨어진다는 단점이 있다.

2.3.2 질의 분할을 통한 병렬화

BLAST의 질의 분할을 통한 병렬화는 클러스터의 각 노드 또는 SMP 장비의 CPU별로 질의 서열을 분할하여 실행하는 방법으로 (그림 2)와 같이 서로 다른 질의를 병렬로 수행할 수 있다. 이 방법은 각 노드의 로컬 디스크나 원격의 공유디스크에 존재하는 데이터베이스에 대하여 분할된 질의를 노드가 개별적으로 수행하게 된다[11,12].

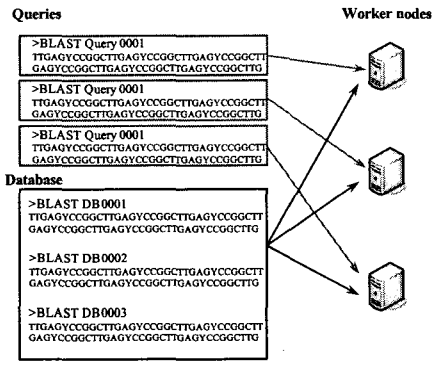

그림 2. 질의 분할
Fig. 2 Query segmentation

2.3.3 데이터베이스 분할을 통한 병렬화

BLAST의 데이터베이스 분할을 통한 병렬화는 (그림 3)과 같이 각 프로세서나 클러스터의 노드 단위로 서열 데이터베이스를 독립적으로 분할하여 검색하는 방법으로 이렇게 함으로써 데이터베이스에서 요구하는 메모리양을 줄일 수 있다. 이러한 구조는 중복성이 배제됨으로써 데이터베이스 양 증가에 따라 컴퓨팅 파워를 추가할 수 있는 장점이 있다. 이러한 데이터베이스를 분할 방법은 TurboWorx사의 TurboBLAST가 있다. TurboBLAST는 워크스테이션과 네트웍을 통한 데이터베이스 분할과 분산 실행을 수행하며, 이때 TurboHub를 통하여 스케쥴링과 로드밸런싱을 수행하며 클러스터 환경에 적용하여 사용할 수 있다[13].

데이터베이스 분할의 또 다른 구현은 parallelblast로 Sun Grid Engine/PVM 환경과 몇 개의 스크립트로 구성되어 있다. 그러나 SGE/PVM 환경을 필요로 한다는 단점이 있다[14].

mpiBLAST는 BLAST 알고리즘 자체는 수정하지 않고, 프로세스 제어와 생성, 데이터 통신에만 수정을 가한 BLAST 프로그램이다. mpiBLAST는 데이터베이스가 분할되어 공유 저장장치에 저장되는 단계와 mpiBLAST 질의가 각 계산 노드에서 검색되는 두 단계로 이루어진다. mpiBLAST의 검색 단계는 검색에 필요한 데이터베이스 파티션 분할 단계와 실제 검색 단계, 그리고 결과 취합 단계로 다시 나뉘어 진다. 즉, 주어진 사용자 질의에 대해 필요한 데이터베이스 파티션을 각 계산 노드로 복사하고, 복사된 각 파티션 내에서 서열의 유사성 검색을 수행한 후 각각의 파티션에 대한 검색 결과를 취합하는 과정을 거친다[15].

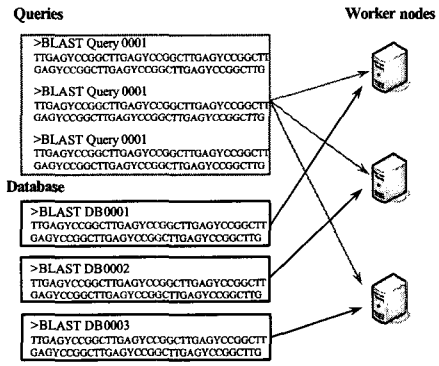

그림 3. 데이터베이스 분할
Fig. 3 Database segmentation

2.4 병렬 프로그래밍 모델

본 절에서는 병렬 프로그래밍을 위한 클러스터와 프로그래밍에 대해서 살펴본다.

2.4.1 리눅스 클러스터

클러스터는 네트워크로 연결된 컴퓨터의 그룹이 하나의 공통된 작업을 처리할 수 있도록 구성된 시스템으로 클러스터에 쓰이는 컴퓨터는 보통 PC에서 워크스테이션에 이르기까지 어떠한 것도 가능하다[16,17]. 이러한 클러스터는 특수한 하드웨어 대신 값싼 일반 컴퓨터에서 사용하는 하드

웨어를 사용하기 때문에 비슷한 성능의 전용 시스템에 비해 비용이 절감된다는 장점이 있다. 또한 이러한 클러스터를 구축하는 운영체제로 오픈 소스를 통해 공개된 리눅스를 사용하여 비용을 절감할 수 있을 뿐만 아니라, 소스코드의 수정이나 커널의 파라미터 값을 수정하여 클러스터 시스템의 성능향상을 기대할 수 있다.

2.4.2 MPI(Message Passing Interface)

클러스터를 구축하는 가장 큰 이유는 복잡한 계산을 여러 노드에 나누어 빠르게 계산하기 위해서다. 이러한 목적을 위해 병렬 프로세서 환경이나 또는 클러스터 환경에서 주로 사용되는 프로그래밍 방식을 병렬 프로그래밍이라고 하며 병렬 프로그래밍 모델은 크게 데이터 병렬 프로그래밍, 메시지 전달 프로그래밍, 쓰레드 프로그래밍으로 분류할 수 있다[18].

데이터 병렬 프로그래밍 모델은 데이터를 분산하여 각 프로세서가 데이터의 특정 영역만을 사용하는 방법으로 이러한 프로그래밍 모델은 데이터 병렬 구조를 가진 프로그램을 작성하고 또한 이 모델을 지원하는 컴파일러로 프로그램을 컴파일 해야 한다. 메시지 전달 프로그래밍 모델은 여러 개의 프로세스가 독립적인 메모리 공간을 사용하여 메시지를 주고받으면서 병렬 계산을 수행하는 방법으로 이 모델은 단일 프로그램이냐, 여러 프로그램이냐에 따라 SIMD(Single Instruction Multiple Data)와 MIMD(Multiple Instruction Multiple Data)로 나누어진다. SIMD는 모든 프로세스가 같은 프로그램을 수행하지만 서로 다른 데이터를 사용하는 방법이며, MIMD는 프로그램과 데이터가 프로세스마다 다른 경우를 말한다.

쓰레드 프로그래밍 모델은 하나의 프로세스 안에 여러 개의 쓰레드를 두어 동시에 여러 작업을 수행할 수 있는 방법으로 이는 운영체제가 쓰레드를 지원해야 하며, 그렇지 않더라도 쓰레드를 지원하는 프로그래밍 언어를 사용해야 한다. 대표적으로 C의 pThread 라이브러리와 자바의 Thread 클래스가 있다[16,17,18].

메시지 전달 프로그래밍 모델을 구현하기 위해서는 메시지 전달 라이브러리가 필요한데, 이것은 프로세스 간에 데이터를 주고받기 위해서 사용된다. PVM[19], MPI, PBS[20]등이 이 라이브러리에 속하며, 가장 많이 사용되는 방법은 MPI(Message Passing Interface)로 클러스터 노드 간 메시지를 서로 주고받아 계산을 병렬적으로 수행한다든지, 데이터의 동기화를 수행한다[18,21].

III. 병렬 서열 분석 디자인 및 알고리즘

본 장에서는 nBLAST의 병렬화를 위한 알고리즘과 노드 정보를 전달하기 위한 메시지 포맷에 대해서 살펴본다.

3.1 MPI 기반 병렬화 아키텍처 디자인

클러스터 환경에서 특별한 조치 없이 MPI 프로그램을 수행하게 되면 프로그램 간 네트워크 사용에 대한 경합 등에 의해 현격한 성능 저하를 유발하게 된다. 따라서 이러한 성능 저하를 방지하기 위해서는 노드간의 부하 분산 방법이 필요하다. 이러한 부하 분산을 위해서는 클러스터의 전체 노드 수, CPU, 메모리 정보 등의 전반적인 정보와 함께 클러스터의 각 노드에 대하여 작업 할당 정보 등을 필요로 한다. 이를 기반으로 노드간의 중첩 없이 균등한 작업 분배를 통해 클러스터상의 성능 저하를 방지 할 수 있다.

본 절에서는 노드들에 대한 정보를 교환하기 위한 메시지 포맷과 마스터 노드와 계산 노드간의 메시지 전달 모델에 대해서 살펴본다.

3.1.1 노드 정보 메시지 포맷

클러스터 환경에서 수행하고자 하는 작업을 각 작업 노드에 분배할 때 load average값이나 CPU 부하에 따라 어떠한 정책을 가지고 작업을 분배하는가에 대한 여러 가지 방식들이 존재한다. 클러스터 환경에서는 Condor, PBS, LoadLeveler, Torqueue 등의 부하 분산 시스템들이 서로 다른 방식의 부하 분산을 제공하고 있다[20].

nBLAST는 마스터 노드와 계산 노드간의 BLAST 작업을 분배하기 위한 방법으로 각 노드에 할당된 노드의 고유번호(Nid), CPU의 개수 정보(Ncpunum), 노드의 CPU의 사용현황(Ncpu), 노드의 1분간 로드 정보(N1load), 노드의 시스템 로드 정보(Nload), 노드의 작업 부여 여부(Nstat)를 필요로 한다. 이를 통해 노드의 작업을 할당 여부를 판단하게 된다.

노드의 고유번호(Nid)는 각 노드를 식별하기 위한 식별자로 사용되며, 시스템 로드 정보(Nload)는 식(2)와 같이 1분 동안의 시스템 로드(Nload)에 대한 CPU 개수(Ncpu num)의 비율로 로드 정보가 50% 미만의 경우에 대해서 안정적으로 작업을 수행 한다고 판단하고 50% 이상에 대해서는 작업을 분배하지 않는다.

Nload(Node_Load_Info) = N1load(1 min load) / Ncpunum(number of CPUs) * 100 ····· 식(2)

현재 노드의 작업 부여 여부(Nstat)는 nBLAST 작업이 노드에 할당되었는지에 대한 정보를 나타낸다. (표 2)는 n BLAST에서 마스터 노드와 작업 노드간의 통신 시 사용되는 메시지에 대한 명세이다.

노드에 대한 정보는 nBLAST가 실제 질의를 분할하는 과정 전에 수행되어 사용 가능한 노드 정보를 수집, 부하 분산을 통해 nBLAST를 수행하게 된다. 또한 노드에 대한 정보는 독립적으로 수행되어 클러스터의 작업 진행 상태 등의 정보를 수집하는데 사용된다.

표 2. nBLAST의 노드 정보 메시지
Table. 2 message of nBLAST

	메시지 내용	데이터 타입
Nid	노드의 고유 식별 번호	Integer
Ncpunum	노드의 CPU 개수	Integer
Ncpu	노드의 CPU 사용량	Integer
N1load	노드의 1분간 로드 정보	Integer
Nload	노드의 시스템 로드 정보 0 : 50% 이상 1 : 50% 미만	Integer
Nstat	노드에 할당된 작업 여부 0 : 작업이 할당되지 않음 1 : 작업이 할당	Integer

3.1.2 메시지 전달 모델

클러스터 환경에서의 nBLAST를 수행하기 위한 마스터 노드와 계산 노드간의 메시지 전달은 MPI 라이브러리를 통하여 송/수신 하게 된다. (그림 4)는 마스터 노드와 계산 노드간 메시지 송/수신 아키텍처를 나타낸다. 마스터 노드는 병렬화를 위한 질의의 분할과 노드 정보를 기반으로 작업을 분배하고, 결과를 취합하게 된다. 계산 노드는 마스터 노드로부터 작업을 할당받아 이를 수행하고, 결과를 전달하는 역할을 하게 된다.

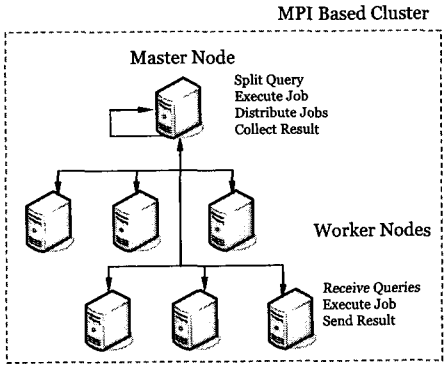

그림 4. 마스터 노드와 계산 노드 아키텍처
Fig. 4 the architecture of master node and worker nodes

3.2 질의 분할 병렬 수행 알고리즘

nBLAST에서의 BLAST 병렬화는 마스터 노드와 계산 노드의 두 부분으로 구분된다. 마스터 노드는 입력된 질의를 분할하고 노드 정보를 통해 얻어진 값을 기반으로 질의를 수행할 노드를 선별하는 과정과 계산 노드에서 실행된 결과를 취합하는 과정을 담당한다. 계산 노드는 마스터 노드로부터 전달된 질의를 수행하고 마스터 노드에 노드 정보 및 질의 결과를 전달한다.

(그림 5)는 nBLAST의 마스터 노드의 알고리즘으로 MPI 통신자의 선언으로 시작해서 MPI 통신자를 해제함으로써 종료된다. 마스터 노드의 작업은 BLAST 작업을 수행할 수 있는 계산 노드가 존재한다는 조건 하에서 시작된다. 계산 노드가 존재한다면 각 계산 노드에 전달할 질의를 분할하게 된다. 여기서 질의 분할 작업은 모든 클러스터 노드에서 진행되는 것이 아니라 마스터 노드에서 이루어지게 된다.

```
질의 서열 집합 Q = {q1; q2; q3...}
할당되지 않은 질의 Unassigned
질의가 수행되고 있지 않은 노드 Unsearched
클러스터 계산 노드 집합 W = {w1; w2; w3...}
질의를 할당받은 계산 노드의 집합 F = {F1; F2; F3...}
통신자에서 자신의 순번(마스터 프로세스 0) iproc
통신자에서 모든 노드의  합 nproc

include MPI Library // MPI 함수

MPI_Init //MPI 통신자를 선언한다.

if |W| ≠ 0 //계산 노드가 존재해야만 한다.
    if iproc = 0 //마스터 노드인 경우
        Split query and save //질의 분할을 수행한다.

    while |Unassigned| ≠ 0 do //모든 노드에게 질의가 할당 될 때까지
        Send a query qi to a worker wj //질의를 계산 노드에 할당한다.
        Remove qi from Unassigned //질의를 할당되지 않은 질의에서 제거한다.
        Add qi to Fj //질의를 할당받은 계산 노드 집합에 추가한다.
    end while

    Receive Broadcast message from workers //계산 노드로부터 질의를 할당 받았다는 메시지를 수신한다.

    while |Unsearched| ≠ 0 do //모든 계산 노드가 질의를 수행할 때까지
        Receive a message from worker wj //계산 노드로부터 질의 결과를 수신한다.
        if message is SEARCH_COMPLETE //계산 종료 메시지를 수신하면
            Receive result //질의 결과를 수신 받는다.
```

```
Remove F, from Unsearched // 질의가 수행되지 않는 노드로
환원한다.
        end while
  if iproc = 0 //마스터 노드인 경우
        Merge the results  //결과 취합
end if

MPI_Finalize //MPI 통신자 해제
```

그림 5. 마스터 노드 알고리즘
Fig. 5 algorithm of master node

nBLAST에서는 MPI를 초기화하는 과정에서 클러스터간의 메시지 송/수신을 위한 통신자를 설정하게 된다. 통신자가 0이라면 이는 작업을 수행하는 마스터 노드를 가리키기 때문에, 통신자가 0인 마스터 노드에서만 분할 작업을 수행하도록 한다. 이 모든 작업은 계산 노드가 존재(시스템 로드 정보(Nload)가 50% 미만인 계산 노드가 존재할 때) 할 때 시작하도록 하여 네트워크의 경쟁 등으로 인한 성능 감소를 사전에 차단한다.

분할 작업 후 생성된 질의는 모든 노드가 질의를 할당 받을 때 까지 계산 노드들에게 질의를 할당하고 질의를 할당 받은 계산 노드는 질의를 수행한 계산 노드 집합에 해당 노드를 추가 시킨다. 모든 질의가 계산 노드에게 할당이 되면 계산 노드들은 BLAST 작업을 수행한다.

```
Recevie query q1 from master node //질의를 할당받는다.
Send back ASSIGN_COMPLETE //질의를 할당 받았음을 통보한다.
Read query and running BLAST //질의를 수행한다.
Send result // 질의 결과를 마스터 노드에 송신한다.
Send back SEARCH_COMPLETE //질의 완료 메시지를 전송한다.
```

그림 6. 계산 노드 알고리즘
Fig. 6 algorithm of worker node

계산 노드의 알고리즘은 (그림 6)으로 마스터 노드로부터 할당 받은 질의에 대하여 질의를 할당 받았음을 통보하고 질의를 수행하게 된다. 수행된 질의가 종료되면 마스터 노드에 질의 결과를 송신하고 질의 완료 메시지를 전송하여 계산 노드의 작업을 완료한다.

IV. 구현

nBLAST는 국립보건연구원 유전체센터의 kgene 클러스터에서 수행되었으며, 시스템 자원은 〈표 3〉과 같다.

표 3. kgene 클러스터의 구성
Table. 3 Specification of kgene cluster

	운영체제	하드웨어
Master (1노드)	Linux (Kernel 2.4.18)	2-way p-III 1.2GHz CPU 1 GB memory
Worker (28노드)	Linux (Kernel 2.4.18)	1-way P-IV 2.0GHz 1 GB memory
Storage (1노드)	Linux (Kernel 2.4.18)	2-way p-III 1.2GHz CPU 1 GB memory

kgene 클러스터는 RedHat Linux 기반의 운영체제를 사용하고 있으며, 총 28노드의 계산노드로 100Mbps의 네트워크 대역폭을 가지고 있다. 스토리지 노드는 모든 계산노드에 NFS로 공유된 마운트 포인트를 제공한다. nBLAST의 바이너리 파일과 질의 파일들은 MPI 환경에서 수행되기 때문에 마스터 노드와 계산 노드의 공유 디스크에 위치한다.

BLAST 데이터베이스의 저장 위치에 대해서는 공유디스크나 계산 노드의 로컬 디스크에 저장할 수 있다. 공유디스크를 이용할 경우 네트워크를 통한 계산 노드간의 부하가 생기지만, 데이터베이스의 업데이트 등의 관리적인 측면에서는 유용하다. 데이터베이스가 각 계산 노드에 존재할 경우 데이터베이스 업데이트 시 동기화의 과정을 필요로 한다는 단점이 있지만 각 계산 노드로 데이터베이스를 로드 하는데 걸리는 네트워크 부하를 줄일 수 있기 때문에 본 논문에서는 각 계산 노드의 로컬 디스크에 BLAST 데이터베이스를 위치시켰다.

(그림 7)은 kgene 클러스터의 모든 노드들에 대한 정보를 Ganglia를 통해 모니터링하고 있는 모습이다. 각 노드들은 현재 CPU나 메모리 등의 사용량 정보를 rrdtool을 이용하여 그래프를 통해 보여준다.

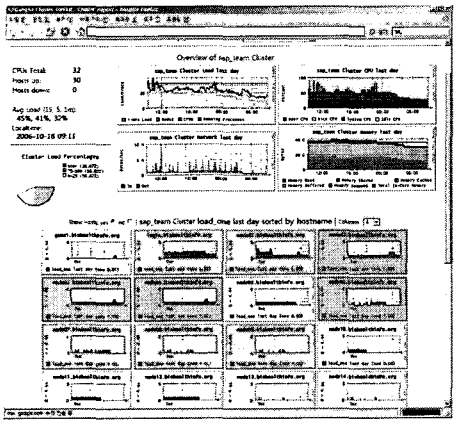

그림 7. kgene 클러스터 모니터링
Fig. 7 kgene cluster state

V. 성능 분석

본 장에서는 실제 nBLAST를 kgene 클러스터에서 계산 노드 수를 변경하면서 측정한 성능 측정값과 nBLAST가 클러스터 상에서 수행될 때의 프로파일러를 통하여 nBLAST 내에서 각 함수가 차지하는 비율을 살펴본다.

5.1 프로파일링 정보

프로파일링은 프로그램 실행 중 프로그램이 어디서 시간을 소비하는지 그리고 어떤 함수를 호출하는지를 보여준다. 따라서 프로파일링 정보를 통해 nBLAST의 병렬 처리 시 그 효율성을 알아보는데 유용하다.

nBLAST의 프로파일링은 GNU gprof를 통해 nBLAST 프로그램 실행 중 자료를 수집한다. nBLAST는 MPI 라이브러리를 사용하는 C언어로 작성되었기 때문에 프로파일링 하기에 앞서 컴파일 과정에서 gcc의 -pg 옵션을 주어 컴파일 한다. -pg 옵션을 통해 생성된 바이너리 파일을 실행하면 프로파일링 정보가 저장된 gmon.out 파일이 생성되고 이를 통해서 nBLAST의 각 함수의 수행시간과 호출횟수 등의 정보를 얻는다.

본 논문에서 nBLAST 알고리즘은 3개의 주요 함수로 구성되며 다음과 같다. (1)querysplit 함수는 질의로 주어진 서열에 대해서 분할 작업을 수행하는 함수이며, (2)exeblast 함수는 분할된 서열에 대해서 순수하게 BLAST를 수행하는 부분이다. 마지막으로 (3)querymerge함수는 각 노드의 결과물의 취합하는 함수이다. 이 함수에 대해서 프로파일링을 통해 실행 시간을 측정한 결과 전체적으로 BLAST를 수행하는 시간에 비해 병렬화 처리를 위한 질의의 분할이나 질의의 취합 부분은 상대적으로 짧은 시간을 필요로 한다. 그러나 노드의 수가 증가함에 따라 MPI 통신을 위한 오버헤드로 인하여 서열 분할 작업과 결과를 취합하는 과정에서의 소요시간이 점차 증가함을 보였다.

5.2 성능 측정

본 논문에서 제안한 nBLAST의 성능을 측정하기 위한 소프트웨어의 구성은 (표 4)와 같다.

표 4. BLAST 환경 설정
Table. 4 Specification of BLAST

BLAST	Version 2.2.9
BLAST DB	nr (994,617 sequences; 194,022,063 total letters)
MPICH	Version 1.2.7p1

BLAST 데이터베이스는 994,617개의 서열로 구성된non-redundant(nr) 단백질 데이터베이스를 사용하였다. 질의 서열의 개수는 64, 256, 1024개에 대해서 계산 노드 수 2~14노드까지 변경하면서 시리얼로 동작하는 BLAST와 성능을 비교하였다. nBLAST의 수행시간과 성능 향상은 (그림 8, 9)와 같다.

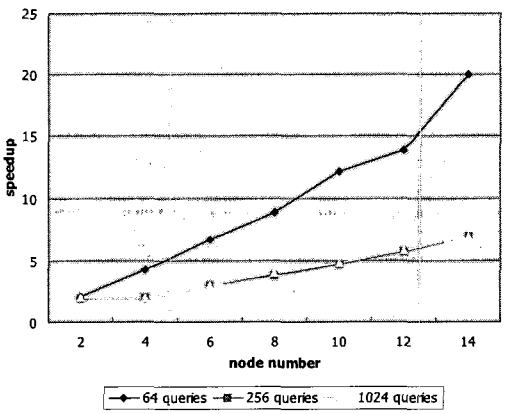

그림 8. 노드 변화에 따른 nBLAST 성능향상
Fig. 8 the speedup of nBLAST

64개의 질의의 경우 노드의 변화에 따라 선형적인 성능 향상을 보이지 않고 있다. 이것은 64개의 질의를 수행하는 시점에서의 MPI 통신에 대한 오버헤드로 인한 문제이다. 질의의 수가 256개로 증가하면서 선형적인 성능 향상을 보

였다. 성능 측정결과 MPI를 사용하여 병렬화된 알고리즘을 통해 64개의 질의를 14노드에서 수행한 결과 2노드에서 수행했을 경우에 비해 9.5배의 성능향상(speedup)을 보여주었다. 이는 단일 BLAST에서 14분의 소요되는 작업을 14노드에서는 1분이라는 시간으로 단축시킬 수 있다.

$$\text{Speedup}(14\text{node}) = \frac{Time(2node)}{Time(14node)} = \frac{885.6(sec)}{90(sec)} = 9.50666667$$

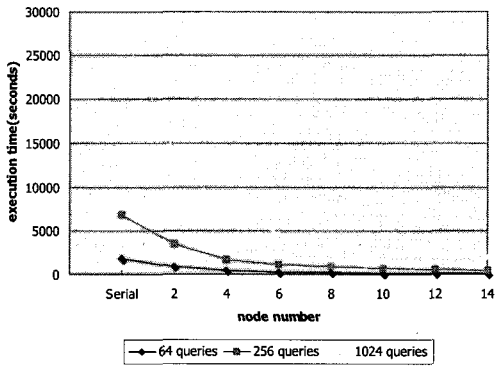

그림 9. 노드 변화에 따른 nBLAST 수행시간
Fig. 9 the performance of nBLAST

노드와 질의수의 변화에 따른 수행시간은 노드 수가 증가 할수록 병렬화 되어 수행되기 때문에 속도가 향상됨을 보였다.

VI. 결론 및 향후 연구

본 논문에서는 BLAST의 병렬화를 위한 MPI 기반의 nBLAST를 제안하였고, 이를 클러스터 환경에서 검증하였다. nBLAST는 BLAST의 병렬화 방법 중에서 질의의 분할을 통하여 다수의 클러스터 노드에 분할된 질의를 수행하도록 하였다. 기존의 병렬화 방법에 비해 별도의 BLAST에 대한 소스코드의 수정이나 부하분산시스템이 필요하지 않아 효과적으로 병렬화를 수행할 수 있도록 하였다.

nBLAST는 단백질 데이터베이스인 nr 데이터베이스에 대해서 blastx를 질의의 수와 클러스터 노드의 수의 변경하면서 성능을 측정하였다. 64개의 질의의 경우 처리 성능은 노드의 수에 대해 비례하여 증가하지 않았다. 이는 질의 수행 시간에 비해 MPI의 오버헤드가 차지하는 비율 때문이며, 1024개의 질의의 경우 노드의 추가에 따라 비례하게 성능이 증가 되었다.

향후 연구로는 클러스터 환경에서의 데이터베이스 병렬화와 노드 정보의 수집을 다른 부하분산 시스템과 연동하여 하나의 클러스터 환경에서 여러 가지 부하분산 방식을 공유하여 계산 노드의 활용도를 높일 뿐만 아니라 노드의 안정성과 클러스터 전체의 가용성을 높일 계획이다.

참고문헌

[1] T.R. Smith and M.S. Waterman. Identification of common molecular subsequences. J. Mol. Biol., 195-197. 1981.

[2] Chemical Engineering Research Information Center, http://www.cheric.org/research/pspdb/

[3] S.Altschul, W.gish, W. Miller, E.Myers, and D.Lipman. Basic local alignment search tool. Journal of Molecular Biology, 215:403,1990.

[4] S.F. Altschul and W. Gish. Local alignement statistics. Methods in Enzymology, 266:460- 480, 1996.

[5] WU-BLAST, http://blast.wustl.edu/

[6] BioGrid, http://www.biogrid.jp

[7] S. Henikoff and J.G. Henikoff. Amino acid substitution matrices from protein block. Proceedings of National Academy of Science, 89:10915- 10919,1992.

[8] Anne Julich. Implementations BLAST for Parallel Cpmputers. CABIOS. 1995.11:3-6.

[9] R. K. Singh, W. D. Dettloff, V. L. Chi, D. L. Hoffman, S. G. Tell, C. T. White, S. F. Altschul, and B. W. Erickson, "BioSCAN: A Dynamically Reconfigurable Systolic Array for Biosequence Analysis," Research on Integrated Systems, 1993.

[10] TimeLogic. Adaptable hardware acclerated systems for bioinformatics. Technical report, TimeLogic, 2002

[11] N. Camp, H. Cofer, and R. Gomperts. High-throughput BLAST, September 1998.

[12] R. Chen, C. Taaffe-Hedglin, N. Willard, and J. Riedl. Efficiency of shared-momory multiprocessors for genetic sequence similarity search algorithm, 1997.

[13] R. D. Bjorson, A. H. Sherman, Weston, N. Willard, and J. Wing. TurboBLAST: A parallel implementation of BLAST based on the TurboHub process integration architecture. Technical report, TurboGenomics, Inc, 2002.
[14] parallelblast, ftp://saf.bio.caltech.edu
[15] A. Darling, L. Carey, and W. Feng. The Design, Implementation and Evaluation of mpiBLAST(Best Paper: Applications Track) ClusterWorld Conference & Expo in conjunction with the 4th Internationl Conference on Linux Clusters: The HPC Revolution 2003, San Jose, CA, June 2003.
[16] Chance Reschke, Thomas Sterling, Daniel Ridge, Daniel Savarese, Donald Becker, and Phillip Merkey A Design Study of Alternative Network Topologies for the Beowulf Parallel Workstation. Proceedings Fifth IEEE International Symposium on High Performance Distributed Computing, 1996.
[17] Thomas Sterling, Donald J. Becker, Daniel Savarese, Michael R. Berry, and Chance Res. Achieving a Balanced Low-Cost Architecture for Mass Storage Management through Multiple Fast Ethernet Channels on the Beowulf Parallel Workstation. Proceedings, International Parallel Processing Symposium, 1996.
[18] P. Pacheco. Parallel Programming with MPI. Morgan Kaufmann. Publisher INC., 1997.
[19] Parallel Virtual Machine, http://www.epm.ornl.gov/pvm/
[20] Portable Batch System, http://www.openpbs.org/
[21] Message Passing Interface, http://www-unix.mcs.anl.gov/mpi/

저 자 소 개

홍창범

2000년 2월 청주대학교 생물학과 졸업
2002년 2월 한남대학교 컴퓨터공학과 공학석사
2004년 1월~ 질병관리본부 국립보건연구원 기술연구원

차정호

2002년 2월 청주대학교 생물학과 졸업
2002년 1월~ 현재 질병관리본부 국립보건연구원 기술연구원

이성훈

1997년 2월 충북대학교 미생물학과 졸업
2001년 8월~ 현재 질병관리본부 국립보건연구원 선임연구원

신승우

1997년 2월 경성대학교 전산통계학과 졸업
1999년 2월 경북대학교 컴퓨터공학과 공학석사
2005년 2월 경북대학교 컴퓨터공학과 공학박사
2006년 1월~ 현재 질병관리본부 국립보건연구원 선임연구원

박근준

1993년 2월 단국대학교 미생물학과 졸업
1997년 3월 교토대학 대학원 이학연구과 석사
2003년 9월 교토대학 대학원 이학연구과 이학박사
2005년 4월 ~ 2006년 3월 도쿄대학 의과학연구소 휴먼게놈센터 특임연구원
2006년 4월 ~ 현재 질병관리본부 국립보건연구원 책임연구원

박근용

1975년 2월 고려대학교 생물학과 졸업
1993년 2월 고려대학교 생물학과 이학석사
1998년 2월 고려대학교 생물학과 이학박사
1992년 4월~ 현재 질병관리본부 국립보건연구원 보건연구관