ISSN(Print): 2287-5905



http://dx.doi.org/10.3745/KTSDE.2013.2.2.107

# Evaluation of Alignment Methods for Genomic Analysis in HPC Environment

Myungeun Lim† · Ho-Youl Jung† · Minho Kim† · Jae-Hun Choi††
Soojun Park††† · Wan Choi†††† · Kyu-Chul Lee†††††

## ABSTRACT

With the progress of NGS technologies, large genome data have been exploded recently. To analyze such data effectively, the assistance of HPC technique is necessary. In this paper, we organized a genome analysis pipeline to call SNP from NGS data. To organize the pipeline efficiently under HPC environment, we analyzed the CPU utilization pattern of each pipeline steps. We found that sequence alignment is computing centric and suitable for parallelization. We also analyzed the performance of parallel open source alignment tools and found that alignment method utilizing many-core processor can improve the performance of genome analysis pipeline.

**Keywords : NGS Analysis, Sequence Alignment, High Performance Computing, Performance Evaluation**

# HPC 환경의 대용량 유전체 분석을 위한 염기서열정렬 성능평가

임 명 은† · 정 호 열† · 김 민 호† · 최 재 훈†† · 박 수 준††† · 최 완†††† · 이 규 철†††††

## 요 약

인간 유전체 지도 완성 후 NGS 기술의 발달로 대용량 유전체 데이터 분석에 대한 요구가 증대하였다. NGS 데이터는 대용량의 단편서열로 구성되므로 효과적인 분석을 위해 고성능 컴퓨팅 기술의 지원이 요구된다. 본 연구에서는 HPC 환경에서 NGS 데이터로부터 SNP를 탐색하는 유전체 분석 파이프라인을 구축하였다. 각 분석 단계의 CPU 이용률 분석을 통해 분석 단계 중 서열 정렬 단계가 연산 작업의 비율이 가장 높은 것을 확인하고, 공개된 병렬화 서열 정렬 도구들의 성능을 분석하여 유전체 분석을 위한 매니코어 프로세서의 활용 가능성을 확인하였다.

**키워드 : NGS 분석, 서열 정렬, 고성능 컴퓨팅, 성능평가**

## 1. 서 론

2003년 약 13년에 걸쳐 인간 유전체 지도의 완성으로 인간 DNA 해독에 대한 관심이 증대되고 다양한 유전체 분석 기술의 출현으로 이어졌다[1]. 하지만 유전체 분석 과정상의 비용 문제로 연구 활성화에 어려움이 있었으나, 차세대 유전체 시퀀싱(NGS, Next-Geneneration Sequencing) 기술의 발달로 이전에 비해 상대적으로 저렴한 가격으로 빠른 시간(2011년 현재 1만달러, 4주)에 유전체 서열 데이터를 획득할 수 있게 되었다. NGS 기술은 개인 수준의 유전체 데이터를 획득하여 분석 가능하게 하며 이를 통해 획득한 정보를 바탕으로 질병, 약물에 관한 정보를 알아내 예방의료 및 개인 맞춤의료의 실현을 앞당길 것으로 기대된다. 이에 따라 NGS 데이터를 중심으로 한 인간 유전체 분석 연구가 활발히 진행되고 있다.

DNA로부터 암호화된 유전정보를 해석하는 일을 유전체 해독이라고 하며, 유전체 분석 기술은 DNA로부터 유전체 상의 변이정보와 같은 유전정보를 해독하기 위한 일련의 기술을 말한다. 유전체 분석을 통해 단일염기다형성(SNP, Single Nucleotide Polymorphism), Indel, 구조적 변이, CNV(Copy Number Variation)등을 찾을 수 있다. 유전체 분석 파이프라인은 유전체 분석 과정에 포함되는 단계적 분석 작업 및 과정을 나열한 것으로, 분석 목적에 따라 서열 정렬, 변이추출, 구조분석과 같은 다양한 작업들로 구성된다.

한편 GPU, MIC와 같은 매니코어(Many core) 프로세서 기술의 등장으로 저비용으로 고성능 컴퓨팅(HPC, High

※ 이 논문은 지식경제부 및 한국산업기술평가관리원의 IT산업원천기술 개발사업의 일환으로 수행하였음[10038768, 유전체 분석용 슈퍼컴퓨팅 시스템 개발].
† 정 회 원 : 한국전자통신연구원 바이오의료IT융합연구부 선임연구원
†† 정 회 원 : 한국전자통신연구원 바이오의료IT융합연구부 책임연구원
††† 정 회 원 : 한국전자통신연구원 바이오의료IT융합연구부장
†††† 정 회 원 : 한국전자통신연구원 클라우드컴퓨팅연구부장
††††† 정 회 원 : 충남대학교 컴퓨터공학과 교수
논문접수 : 2013년 1월 8일
수 정 일 : 1차 2013년 1월 24일
심사완료 : 2013년 1월 24일
* Corresponding Author : Kyu-Chul Lee(kclee@cnu.ac.kr)

Performance Computing) 환경의 구축이 용이해졌다. 대표적인 매니코어 프로세서인 Nvidia의 Tesla K20X GPU는 2688개 코어를 내장하여 1.31 Tflops의 연산 성능을 내며, 인텔의 Xeon Phi 5110P 코프로세서는 61개 코어로부터 1.01 TFlops의 성능을 보유한다. 고성능 및 확장의 용이성 등으로 매니코어 제품은 케이, 티엔허1-A, SGI UV2000 등의 슈퍼컴퓨팅 환경 구축에 사용되고 있다.

매니코어 프로세서 기술은 최근 유전체 연구 분야에도 도입되어 대용량의 유전체 데이터 분석에 활용되고 있다. 서열분석 기술에는 GPU를 기반으로 한 CUDA-SW++, Barracuda, SOAP3 등[2-4]이 개발되었고, 생체분자학 시뮬레이션을 위한 Amber-GPU[5] 등이 사용 중에 있으며 점점 활용 범위가 넓어지고 있는 추세이다.

NGS 분석은 대용량의 데이터를 다루므로, 이를 처리하기 위한 고성능 컴퓨팅 기술의 지원이 필수이다. 이를 위해 ETRI에서는 2011년부터 유전체 분석을 위한 고성능 컴퓨팅 기술을 개발하고 있다. 본 논문에서는 현재 공개된 유전체 분석 도구를 바탕으로 매니코어 기반의 고성능 컴퓨팅 환경에서 인간 유전체를 분석하기 위한 유전체 분석 파이프라인에 대해 소개하고, 대량의 연산을 필요로 하는 서열 정렬 단계의 성능 향상을 위해 공개된 서열 정렬 시스템들의 성능을 측정하여 장단점을 분석한다.

본 논문의 구성은 다음과 같다. 2장에서 유전체 분석을 위한 분석도구들에 대해 소개하고, 3장에서 구축된 HPC 기반의 유전체분석 파이프라인에 대해 설명한다. 4장에서 서열정렬기법들에 대한 성능평가결과를 분석하고, 5장에서 결론으로 끝을 맺는다.

## 2. 관련 연구

### 2.1 서열 정렬

서열 정렬(sequence alignment)이란 참조서열에서 주어진 DNA 서열의 위치를 찾아내는 과정을 말한다. 유사한 뉴클레오티드 또는 아미노산 서열은 화학적 특성으로 인해 유사한 구조를 가지며 이로 인해 유사한 생물학적 기능을 가지는 특징이 존재하므로 서열로부터 구조와 기능적인 정보를 밝히기 위해 서열 정렬을 사용한다. NGS 기술은 전사체 연구, RNA 연구, 변이탐색, 메타게놈, 메틸레이션 분석 등의 다양한 분야에 기여하고 있는데, 이런 연구들의 가장 선행 작업이 서열정렬 과정이다.

전통적인 서열 정렬 방법으로는 Needleman-Wunsch, Smith-Waterman 알고리즘[6-7]과 같은 동적 프로그래밍(dynamic programming) 기법이 있다. 이 방법은 서열의 정렬 스코어 계산에 있어 이전 셀의 스코어를 활용하여 특정 염기 또는 아미노산의 유사도를 계산하는 방식이다. Needleman-Wunsch[6]는 비교 염기 불일치 시 음의 스코어를 가지는 계산 방식으로 두 서열의 전체 유사도를 검색하는 전역정렬(glogal alignment)에 적합하고, Smith-Waterman[7]은 불일치 시 0의 스코어를 가지는 계산 방식으로 두 서열 내에서 가장 유사한 서열 구간을 찾는 지역정렬(local alignment)에 사용된다. 동적 프로그래밍 기법은 서열정렬을 위해 전체서열을 모두 비교하므로 서열 길이에 비례하여 정렬 시간이 매우 오래 걸리는 단점이 있다.

동적 프로그래밍 기법의 수행속도를 개선하기 위해 'seed-and-extend' 개념을 적용한 BLAST[8]가 개발되었다. BLAST는 입력서열로부터 추출한 시드로 해시 테이블을 작성하고 이것을 데이터베이스에 검색한 후 유사도가 높은 시드에 대해 매핑을 확장하는 방식으로 정렬한다. BLAST는 동적 프로그래밍 방식에 비해 50배 이상의 빠른 수행 성능을 보이지만 최적의 해를 보장하지 않는 단점이 있다.

### 2.2 NGS 서열 정렬

한편 Illumina의 GA, Loche/454의 GS FLX, ABI의 SOLiD 등 최근의 차세대 시퀀싱 플랫폼에서 생산되는 서열 데이터는 길이가 35-400bp(base pair, 염기서열의 개수 단위)로 Sanger 방식의 650-800bp에 비해 상대적으로 서열의 길이가 짧고, Reading 정확도 및 Base Calling 신뢰도가 낮은 것이 특징이다. 이런 특징을 감안하여 NGS 기기에서는 일반적으로 15-50배 정도의 중복된 서열 데이터를 생산하는데 이 때문에 저장공간, 메모리, 정렬 속도에 대한 최적화 문제가 발생한다.

NGS를 위한 서열 정렬 방법은 해시 또는 접미사트리(Suffix tree) 등의 색인을 이용한다. MAQ[9]는 해시를 이용하는 대표적인 시스템으로, 입력 서열들로부터 해시 테이블을 생성하고 이것을 참조서열에 매핑하는 방식으로 정렬을 수행한다. 이러한 방식은 개별 서열단위의 멀티스레딩이 불가능하고, 처리할 입력 서열이 많을 경우 다량의 메모리를 필요로 한다. 반면에 BFAST, SOAP 등[10-11]은 참조서열을 색인하는 방법을 사용하여 멀티스레딩이 가능하나 역시 해시테이블 구성에 많은 메모리를 요구한다.

BWA, Bowtie, SOAP2 등[12-14]은 참조서열에 대해 BWT(Burrows Wheeler Transform) 기반의 색인을 구성하여 정렬에 활용한다. 일반적으로 접미사트리 기반 알고리즘은 색인유지에 많은 양의 메모리를 요구하지만 BWT 기반의 방식들은 참조서열 크기에 비해 적은 양의 색인 공간(인간 유전체의 경우 약 2.3GB)을 사용하면서도 빠른 정렬 성능을 보인다. BWT 기반의 서열 정렬 기법들에서 완전일치 검색은 BWT기반 색인에 후방검색을 수행하는 방식으로 거의 유사하게 이루어지며 갭(gap), 미스매치(mismatch) 처리 방법에 따라 각기 특징을 가진다. BWA[12]는 힙구조의 탐색후보군 버퍼를 유지하여 삽입/삭제 발생 시 후보군에 대해 정렬을 계속하는 방식으로 갭을 허용한다. 반면 Bowtie[13]는 BWT 탐색 시 갭은 허용하지 않으며, 염기 치환과 재검색을 통한 백트래킹으로 미스매치 검색을 처리한다. SOAP2[14]는 허용되는 미스매치의 수에 따라 서열 영역을 나누고 분리된 영역에 대해 양방향으로 정렬을 수행하는 양방향(Bi-directional) BWT 탐색 방법을 사용한다.

### 2.3 병렬화 기법을 적용한 서열 정렬

대용량 NGS 데이터에 대한 서열 정렬 시간을 단축하기 위해 프로세서의 병렬 구조를 활용하는 방안들도 다수 연구되고 있다.

Farrar는 Intel CPU의 SIMD 아키텍처를 활용하여 Smith-Waterman 알고리즘을 병렬 실행하는 방법을 제안하였다[15]. Smith-Waterman 알고리즘은 각 셀의 스코어가 이전 셀에 종속적이어서 연속된 셀들의 병렬처리가 불가능하다. Farrar는 연속된 셀 대신 종속성이 없는 일정 간격의 셀들의 집합을 SIMD 명령으로 처리가 가능한 벡터 세그먼트로 묶어 병렬적으로 스코어를 계산 후 잘못된 셀에 대해서만 스코어를 갱신하는 lazy-F 연산을 통해 종속성 문제를 해결하고 속도를 개선하였다. Bowtie2[16]는 단편서열에서 시드를 추출하여 BWT 색인에 완전일치 검색을 실행하고 검색된 시드에 대해 SIMD 벡터화한 동적 프로그래밍으로 입력서열을 확장하는 'seed-and-extend' 방식의 정렬을 수행한다.

한편 GPU의 확산에 따라 기존의 서열정렬 알고리즘을 GPU 상에서 응용하는 사례도 늘고 있다. CUDA-SW++[2]는 GPU 코어 기반의 멀티스레드로 Farrar의 SIMD의 벡터 모델을 가상화하여 GPU에서 구동 가능하도록 수정 적용하였다. BarraCUDA[3]는 BWA를 GPU에 이식한 버전으로, BFS 기반의 후방검색 방식을 DFS 방식으로 변경하여 GPU 스레드 당 메모리 사용량을 줄이고 GPU 커널 함수를 분할하여 GPU 상에서 효과적으로 동작하도록 하였다. SOAP3-dp[4] 역시 'seed-and-extend' 개념을 기초로 GPU에서 BWT 색인에 따라 1차 매핑 후 실패 시 동적 프로그래밍을 수행하는 형태로 정렬을 실행한다. 또한 다중 GPU를 지원하여 2개 이상의 GPU가 장착된 경우에 장치 활용도를 높인다.

살펴본 바와 같이 사용목적 및 방식에 따라 매우 다양한 형태의 정렬 기법들이 존재한다. 따라서 유전체 분석 파이프라인 구성을 위해서는 각 프로그램에서 지원하는 입출력 형태 및 프로세서의 종류를 고려하여 적절한 정렬 방식을 선택해야 한다.

본 논문에서는 HPC 기반의 유전체 분석 파이프라인 구성을 위한 서열정렬 기법 중 CPU 기반 정렬에서 가장 우수한 성능을 보이는 BWA와, BWA 알고리즘을 GPU로 이식한 Barracuda, 그리고 GPU 기반의 빠른 수행속도를 보이는 SOAP3-dp에 대한 성능을 비교 분석하고자 한다.

### 2.4 SAM Tools

SAM(Sequence Alignment/MAP) 형식은 대량의 뉴클레오티드 서열 정렬 결과를 저장하기 위한 표준 형식이다[17]. SAM 파일에는 질의/참조 서열명, 매핑정보, quality 정보 및 사용자 주석 등 유전체 분석에 필요한 다양한 정보들을 포함한다. 대부분의 서열 정렬 도구들이 SAM 형식의 결과를 출력하거나 변환 방법을 제공한다. BAM 형식은 빠른 연산을 위해 SAM 형식을 이진 형식으로 압축한 것이다.

SAM Tools는 SAM/BAM 형식으로 된 DNA 서열 정렬 결과를 조작하기 위한 다양한 도구의 집합이다. 매핑 결과의 정렬, 병합, 색인, BAM 형식 변환, 변이 추출(variant calling) 등의 기능을 지원한다. 'view'는 정렬결과로부터 SAM/BAM 형식의 데이터를 추출하고, 'sort'는 참조서열 내의 위치를 기준으로 BAM 형식의 매핑 결과를 정렬한다. 'index'는 SAM/BAM 파일내의 빠른 탐색을 위한 색인을 생성하며, 'merge'는 여러 BAM 파일을 병합한다. 'mpileup'은 BAM 파일로부터 BCF(Binary Call Format) 형태로 변이 정보를 추출하며 'BCFtools'는 BCF, VCF 파일을 변환하며 변이 후보를 추출한다. 이처럼 SAM Tools는 유전체 변이 분석에 필요한 다양한 도구를 패키지 형태로 제공하므로 유전체 분석에 필수적인 요소이다.

## 3. HPC 기반 유전체 분석 파이프라인

### 3.1 파이프라인의 구성

본 연구의 유전체 분석 파이프라인은 Fig. 1과 같이 NGS 서열집합으로부터 SNP 정보를 추출하기 위한 일련의 과정들로 구성되며, 크게 NGS 서열 정렬 과정과 SNP 추출(Calling) 과정으로 나뉜다. NGS 서열 정렬 과정은 참조서열을 색인하고 단편서열을 참조서열에 매핑하며, 매핑 결과를 SAM 형식으로 변환하는 과정을 포함한다. 서열정렬 방법은 CPU 기반의 매핑 신뢰도가 높은 BWA와 수행시간 성능이 좋은 SOAP3-dp를 선택적으로 사용할 수 있다. 각 정렬 프로그램들은 정렬 결과를 SAM 형식으로 변환하는 과정을 포함한다. 이후, SAM Tools를 이용해 염색체 별로 매핑 결과를 정렬하고 'flagstat' 명령으로 정렬 결과에 대한 통계정보를 생성한다. SNP 추출 과정은 정렬 결과에 대한 BAM 파일을 병합하고 색인을 생성한 후 'mpileup'을 통해 이진 형태의 변이파일(bcf)을 생성하고 'bcftools'를 이용하여

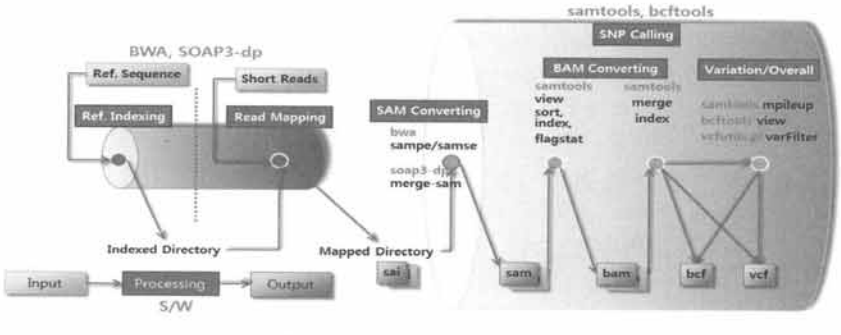


Fig. 1. NGS Analysis Pipeline

```
#!/bin/bash
REF=/home/hyj/Data/index/chr21/chr21.fa

echo "========================= 0. begin bwa index"
time bwa index -a bwtsw $REF
time samtools faidx $REF
echo "========================= 0. end bwa index"

echo "========================= 1. begin bwa aln"
echo "========================= 1.1.1. begin bwa aln read1_1"
time bwa aln -t 15 $REF read1_1.fq > read1_1.sai
echo "========================= 1.1.1. end bwa aln read1_1"
echo "========================= 1.2.1. begin bwa aln read2_1"
time bwa aln -t 15 $REF read2_1.fq > read2_1.sai
echo "========================= 1.2.1. end bwa aln read2_1"
echo "========================= 1. end bwa aln"

echo "========================= 2. begin bwa sampe"
echo "========================= 2.1. begin bwa sampe read1"
time bwa sampe $REF read1_1.sai read1_2.sai read1_1.fq read1_2.fq > read1.sam
echo "========================= 2.1. end bwa sampe read1"
echo "========================= 2. end bwa sampe"

echo "========================= 3. begin samtools sort"
echo "========================= 3.1. begin samtools sort"
time samtools view -Su read1.sam | samtools sort - read1
echo "========================= 3.1. end samtools sort"
echo "========================= 3. end samtools sort"

echo "========================= 4. begin samtools index"
echo "========================= 4.1. begin samtools index"
time samtools index read1.bam
echo "========================= 4.1. end samtools index"
echo "========================= 4. end samtools index"

echo "========================= 5. begin samtools flagstat"
echo "========================= 5.1. begin samtools flagstat"
time samtools flagstat read1.bam > read1.bam.flagstat
echo "========================= 5.1. end samtools flagstat"
echo "========================= 5. end samtools flagstat"

echo "========================= 6. begin samtools merge&index"
time samtools merge read_merged.bam read1.bam read2.bam
time samtools index read_merged.bam
echo "========================= 6. end samtools mege&index"

echo "========================= 7. begin samtools mpileup variation"
time samtools mpileup -uf $REF read_merged.bam | bcftools view -bvcg - > read.bcf
time bcftools view read.bcf | vcfutils.pl varfilter -d 5 -D 150 > read.vcf
echo "========================= 7. end samtools mpileup variation"

echo "========================= 8. begin samtools mpileup overall"
time samtools mpileup -C50 -gf $REF read_merged.bam > read.all.bcf
time bcftools view read.all.bcf > read.all.vcf
echo "========================= 8. end samtools mpileup overall"
```

Fig. 2. NGS Analysis Pipeline Script

최종결과인 변이추출파일(vcf)을 생성한다.

파이프라인의 실행은 대부분 쉘 스크립트 명령으로 구성되어 순차적으로 수행된다. Fig. 2는 쉘 명령 예이다.

### 3.2 유전체 분석 도구 관리 인터페이스

본 연구의 유전체 분석 작업을 효과적으로 관리하기 위해 프로젝트 단위로 작업을 구성하고 실행할 수 있는 관리 인터페이스를 개발하였다. Fig. 3의 유전체 분석 도구 관리 인터페이스를 통해 파이프라인 단계 구성, 단계별 분석 도구 지정, 분석 도구별 파라미터 설정, 입출력 파일 등을 설정할 수 있다. 한편, 다중노드로 구성된 서버의 활용을 위해 서열 정렬 및 SAM 변환 과정 실행 시 복수개의 입력 파일을 가용 노드별로 할당하여 파일 단위의 프로세스 병렬화를 실행한다. 분석 작업이 끝나면 전체 및 노드별 수행시간, 정확도 등에 대한 통계 확인이 가능하다.

## 4. 실험 및 성능측정

유전체 분석 파이프라인의 작업 패턴 분석을 위해 CPU 이용률(utilization)을 모니터링 하는 'iostat'을 이용하여 주요 단계의 CPU 이용률을 측정한 결과는 Fig. 4와 같다. 35bp의 3,615,487개 단편서열들 450MB로 구성된 BGI(Beijing Genome Institute) 데이터에 대해 (a)참조서열 색인, (b)리드 매핑, (c)SAM 형식 변환, (d)BAM 정렬 단계의 CPU 이용률을 측정하였다. Idle 상태를 제외한 CPU 이용률 평가 결과 색인 및 매핑 단계는 입출력에 비해 CPU 연산이 높은 비율을 차지하고, SAM 변환과 BAM 정렬 단계는 입출력 작업이 상대적으로 많은 부분을 차지하는 것으로 나타났다. 이는 전체 파이프라인 중에서 서열 정렬 단계의 경우 입출력 보다 연산 수행이 많아 이를 병렬처리 할 경우 수행시간 개선 효과가 크게 나타날 수 있음을 의미한다.

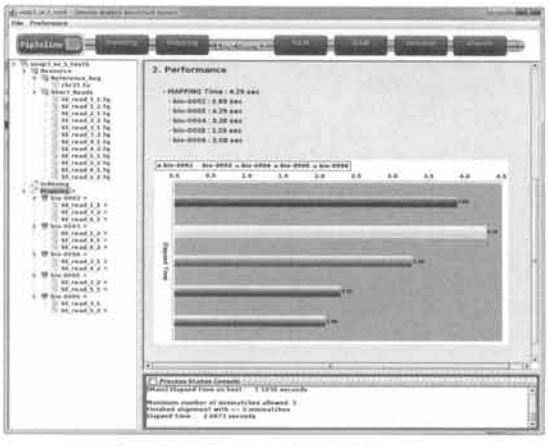

Fig. 3. NGS Analysis Job Manager

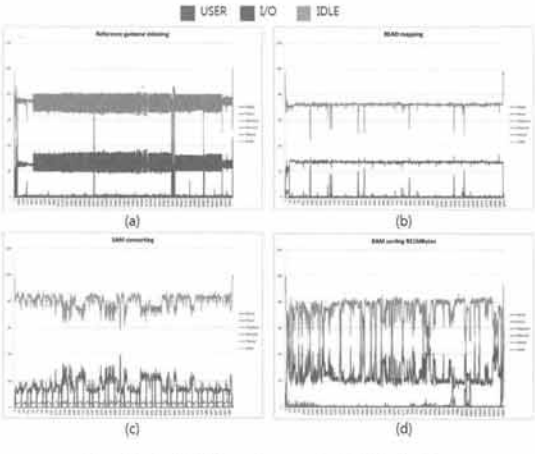


Fig. 4. CPU Utilizations of NGS Pipeline

### 4.1 실험환경

서열의 길이와 전체 데이터양에 따른 서열 정렬의 성능 변화를 측정하기 위해 SAM Tools에서 제공하는 wgsim 도구를 이용하여 Table 1과 같이 시뮬레이션 서열 데이터를 생성하였다. 서열 길이는 50bp, 100bp, 150bp이며 총 염기서열의 길이는 각각에 대해 200Mbp와 2Gbp 두 가지 크기로 구성된다.

그리고, 실제 인간 유전체 데이터에 대한 성능 측정을 위해 PGI(Personal Genome Institute)의 Human Genome 데이터를 사용하였다. 이 데이터는 7개 레인에서 추출된 90bp의 7개 염기 쌍(paired-end) 중 6쌍으로 Table 2와 같이 구성된다. 한편 실험데이터를 정렬하기 위한 참조서열은 2.93GB의 NCBI hg19 유전체 데이터를 사용하였다.

실험에 사용된 서버는 Intel Xeon E5640 2.67GHz, 48G 메모리와 2개의 2TB HDD로 구성되었으며 GPU는 NVidia Tesla C2070을 2기 장착하고 있다.

Table 1. Simulation Dataset

| | 50_2 | 100_1 | 150_0.66 | 50_20 | 100_10 | 150_6.6 |
|---|---|---|---|---|---|---|
| Num. of Seqs. | 4,000,000 | 1,999,994 | 1,333,338 | 40,000,002 | 20,000,002 | 13,333,338 |
| Seqs. Length | 50 | 100 | 150 | 50 | 100 | 150 |
| Total Length | 200Mbp | 200Mbp | 200Mbp | 2Gbp | 2Gbp | 2Gbp |

Table 2. Human Genome Data of PGI

| | s_1 | s_2 | s_3 | s_4 | s_5 | s_6 |
|---|---|---|---|---|---|---|
| Num. of Seqs. | 169,952,352 | 159,592,048 | 154,384,412 | 160,317,272 | 146,770,084 | 137,709,818 |
| Seqs. Length | 15,295,711,680 | 14,363,284,320 | 13,894,597,080 | 14,428,554,480 | 13,209,307,560 | 12,393,883,620 |
| File Size | 35.4GB | 33.2GB | 32.1GB | 33.4GB | 30.5GB | 28.7GB |

### 4.2 서열 정렬 성능 분석

NGS 서열 정렬 평가를 위해 BWA 0.6.1, Barracuda 0.6.2, SOAP3-dp 2.2를 사용하였으며, Table 3은 시뮬레이션 데이터에 대한 시스템 별 수행시간 및 매핑율을 측정한 결과이다. BWA는 16개의 멀티스레드로 실행하였고, SOAP3-dp는 단일GPU를 사용하였다. 수행시간 면에서 BWA는 50bp일 때 132초에서 150bp일 때 142초로 증가한 반면, SOAP3-dp는 152초에서 118초로, Barracuda도 439에서 215초로 감소하였다. 총 서열의 크기가 동일한 상황에서 BWA의 수행시간은 정렬대상 서열의 개수보다 서열의 길이에 영향을 더 받는 반면, Barracuda와 SOAP3-dp는 서열의 개수에 더 영향을 받는 것으로 판단된다. 한편, 서열의 길이가 짧고 총 서열 크기가 작을 경우 CPU 기반의 BWA가 빠르나, 서열 길이가 길고 총 서열의 크기가 클 경우 GPU 기반의 SOAP3 및 Barracuda가 더 빠르거나 개선되는 속도를 보이는데, 이는 GPU 초기화 및 색인 복사 등으로 인한 오버헤드가 작은 데이터에서는 수행시간에 큰 비중을 차지하기 때문으로 판단된다. 매핑율은 세 시스템에서 98% 이상의 성능을 보였으며, BWA의 경우 99% 이상으로 근소하게 안정적인 성능을 나타냈다.

Table 3. Experimental Result of Simulation Data

| | | 50_2 | 100_1 | 150_0.66 | 50_20 | 100_10 | 150_6.6 |
|---|---|---|---|---|---|---|---|
| bwa | Time(sec) | 132 | 140 | 142 | 1202 | 1395 | 1403 |
| | Mapping rate | 99.10 | 99.51 | 99.33 | 99.55 | 99.69 | 99.55 |
| barracuda | Time | 439 | 278 | 215 | 4405 | 2741 | 2182 |
| | Mapping rate | 99.48 | 99.41 | 99.13 | 99.48 | 99.42 | 99.13 |
| soap3-dp | Time | 152 | 128 | 118 | 615 | 429 | 346 |
| | Mapping rate | 98.27 | 98.84 | 99.11 | 98.26 | 98.83 | 99.11 |

Table 4는 17GB의 PGI s_1 데이터에 대한 정렬시간 측정 결과이다. BWA의 경우 멀티스레딩 효과 분석을 위해 스레드 수에 따른 수행시간 변화를 측정하였다. 그 결과 16개의 스레드를 사용한 BWA가 229분으로 최소 수행시간을 보였으며, 이후에 스레드 증가에 따른 시간 변화가 미미하여 16 스레드일 때 최대의 수행 성능을 보이는 것으로 판단된다. Barracuda는 단일 스레드를 사용한 BWA에 비해 2.34배 빠른 수행 성능을 보였고, SOAP3-dp는 단일 스레드 BWA에 비해 약 21.57배, 최적의 멀티스레드 BWA에 대해 4.77배 빠른 수행 성능을 보였다. Barracuda는 서열 정렬에 GPU만을 사용하는 반면, soap3-dp는 CPU와 GPU를 모두 활용하여 높은 성능을 나타내는 것으로 판단된다.

사용되는GPU 개수에 따른 성능의 변화를 측정하기 위해 6개 파일을 단일 GPU를 이용하여 순차적으로 정렬한 시간과 3개씩 2개 GPU에 동시에 정렬한 결과의 실행시간(SAM 변환시간 제외)은 Table 5와 같다. 다중 GPU 사용 시 개별 파일의 실행시간은 증가하지만 매핑을 병렬로 수행하는 효과로 인해 약 1.55배의 실행시간 단축 효과를 보였다.

Table 4. Experimental Result of PGI s_1 Data

| | bwa | | | | barracuda | soap3-dp |
|---|---|---|---|---|---|---|
| Num. of Thread | 1 | 8 | 16 | 32 | | |
| Time(min) | 1057 | 281 | 229 | 226 | 450 | 49 |
| Performance ratio | 1 | 3.76 | 4.61 | 4.67 | 2.34 | 21.57 |

Table 5. Experimental Result of SOAP3-dp using Multi-GPU

| Num. of GPU | Time(min) | | | | | | Total |
|---|---|---|---|---|---|---|---|
| | s_1 | s_2 | s_3 | s_4 | s_5 | s_6 | |
| 1 | 30 | 33 | 44 | 44 | 29 | 31 | 213 |
| 2 | 42 | 40 | 53 | 47 | 41 | 38 | 137 |

## 5. 결 론

본 연구에서는 고성능 컴퓨팅 환경에서 대용량 유전체 분석을 위한 유전체 분석 파이프라인을 구축하고 파이프라인의 각 단계별 작업 성향을 분석하였다. 그 결과 서열정렬 단계의 CPU 이용률이 타 작업에 비해 높아 병렬화에 적합한 것으로 분석되었으며, 이를 바탕으로 공개 서열 정렬 시스템의 성능을 분석한 결과 매니코어 프로세서를 활용한 서열 정렬 기법이 가장 빠른 성능을 보이고 있는 것으로 파악되었다. 이를 통해 유전체 분석 분야에서 매니코어 기술의 활용 가능성을 확인할 수 있었으며, ETRI에서 개발 중인 이기종 매니코어 프로세서 기반의 슈퍼컴퓨팅 시스템이 인간 유전체 분석에 기여할 수 있을 것으로 사료된다.

현재 전체 파이프라인 중 서열정렬 단계만 병렬화 기법이 적용된 상황이나 많은 연산이 요구되는 단일염기다형성 정보 추출과정 등에도 매니코어 및 다중코어 기반의 병렬화 기술을 적용한다면 보다 빠른 시간에 유전체 분석 결과를 얻을 수 있을 것이다.

## 참 고 문 헌

[1] Human Genome Project, http://www.ornl.gov/sci/tech resources/Human_Genome/home.shtml

[2] C. Angermüller, A. Biegert, J. Söding, "Discriminative modelling of context-specific amino acid substitution probabilities," *Bioinformatics*, Vol.28, pp.3240-3247, 2012.

[3] P. Klus, S. Lam, D. Lyberg, MS Cheung, G. Pullan, I. McFarlane, GSH Yeo, BY Lam, "BarraCUDA - a fast short read sequence aligner using graphics processing units," *BMC Research Notes*, Vol.5, 27, 2012.

[4] C. Liu, T. Wong, E. Wul, R. Luo, S. Yiu, Y. Li, B. Wang, C. Yu, X. Chu, K. Zhao, R. Li, T. Lam,, "SOAP3: Ultra-fast GPU-based parallel alignment tool for short reads," *Bioinformatics*, Vol.28, pp.878-879, 2012.

[5] A. Goetz, M. Williamson, D. Xu, D. Poole, S. Grand, R. Walker, "Routine microsecond molecular dynamics simulations with AMBER - Part I: Generalized Born," *Chemistry Theory Computrmatics Journal*, Vol.8, pp.1542-1555, 2012.

[6] B. Needleman, D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, Vol.48, pp.443-453, 1970.
[7] F. Smith, S. Waterman, "Identification of Common Molecular Subsequences," *Journal of Molecular Biology*, Vol.147, pp.195-197, 1981.
[8] S. Altschul, W. Gish, W. Miller, E. Myers, D. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, Vol.215, pp.403-410, 1990.
[9] H. Li, J. Ruan, R. Durbin, "Mapping short DNA sequencing reads and calling variants using mapping quality scores," *Genome Researchl*, Vol.18, pp.1851-1858, 2008.
[10] N. Homer, B. Merriman, SF. Nelson, "BFAST: an alignment tool for large scale genome resequencing," *PLoS One*, Vol.4, e7767, 2009.
[11] R. Li, Y. Li Y, K. Kristiansen, J. Wang, "SOAP: short oligonucleotide alignment program," *Bioinformatics*, Vol.24, pp.713-714, 2008.
[12] Li H. and Durbin R, "Fast and accurate short read alignment with Burrows-Wheeler Transform," *Bioinformatics*, Vol.25, pp.1754-1760, 2009.
[13] B. Langmead , C. Trapnell, M. Pop, SL. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biology*, Vol.10, R25, 2009.
[14] R. Li, C. Yu, Y. Li, TW. Lam, SM. Yiu, K. Kristiansen, J. Wang, "SOAP2: an improved ultrafast tool for short read alignment," *Bioinformatics*, Vol.25, pp.1966-1967, 2009.
[15] M. Farrar, "Striped Smith-Waterman speeds database searches six times over other SIMD implementations," *Bioinformatics*, Vol.23, pp.156-161, 2007.
[16] B. Langmead, S. Salzberg, "Fast gapped-read alignment with Bowtie 2," *Nature Methods*, Vol.9, pp357-359, 2012.
[17] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, "The Sequence Alignment/Map format and SAMtools," *Bioinformatics Journal*, Vol.25, pp.2078-2079, 2009.

**임 명 은**

e-mail : melim@etri.re.kr
1999년 동국대학교 컴퓨터공학과(학사)
2001년 동국대학교 컴퓨터공학과(석사)
2001년~현 재 한국전자통신연구원 바이오의료IT융합연구부 선임연구원
관심분야 : 생물정보학, 기계학습, 병렬처리

**정 호 열**

e-mail : hoyoul.jung@etri.re.kr
1997년 부산대학교 전자계산학과(학사)
1999년 부산대학교 전자계산학과(석사)
2002년 부산대학교 전자계산학과(이학박사)
2002년~2004년 국립보건연구원 유전체센터 책임전문연구원
2004년~현 재 한국전자통신연구원 바이오의료IT융합연구부 선임연구원
관심분야 : 생물정보학, 계산이론

**김 민 호**

e-mail : kimmh@etri.re.kr
1997년 고려대학교 전자공학과(학사)
1999년 광주과학기술원 정보통신공학과(석사)
2006년 광주과학기술원 정보통신공학과(박사)
2006년~현 재 한국전자통신연구원 바이오의료IT융합연구부 선임연구원
관심분야 : Bioinformatics, Machine Learning, Artificial Intelligence

**최 재 훈**

e-mail : jhchoi@etri.re.kr
1994년 전북대학교 전자계산학(학사)
1996년 전북대학교 전산통계학과(석사)
2000년 전북대학교 전산통계학과 (Ph.D., 박사)
2000년~현 재 한국전자통신연구원 바이오의료IT융합연구부 책임연구원
관심분야 : Medical Data Mining, Big Data, Cloud Computing

**박 수 준**

e-mail: psj@etri.re.kr
1991년 University of Iowa, 생화학(학사)
1994년 Lehigh University, 컴퓨터과학(석사)
2011년 동국대학교 전자공학 (박사)
1994년~현 재 한국전자통신연구원 바이오의료IT융합연구부장
관심분야 : 디지털영상처리, 유헬스, 바이오인포매틱스, 데이터마이닝

**최 완**

e-mail : wchoi@etri.re.kr
1981년 경북대학교 전자공학과(학사)
1985년 KAIST 전산학과(석사)
1985년~2003년 ETRI, TDX/CDMA 전전자교환기용 실시간 OS/DBMS/MW/컴파일러 개발책임자
2000년~2011년 한국정보처리기술사회 이사
2004년~2007년 ETRI, 클라우드서비스 기술 개발팀장
2008년~2010년 ETRI, SW콘텐츠미래기술연구부장
2011년~현 재 한국전자통신연구원 클라우드컴퓨팅연구부장
관심분야 : Cloud Computing, High Performance Computing

**이 규 철**

e-mail : kclee@cnu.ac.kr
1984년 서울대학교 컴퓨터공학과(학사)
1986년 서울대학교 컴퓨터공학과(석사)
1990년 서울대학교 컴퓨터공학과(박사)
1989년~1994년 IBM Almaden Research Center, 초빙 연구원
1995년~1996년 Syracuse University, CASE Center, 초빙 교수
1997년~1998년 교육구 학술진흥재단 부설 첨단학술센터, 파견 교수
1989년~현 재 충남대학교 컴퓨터공학과 교수
관심분야 : XML, 정보통합, 유비쿼터스 웹 서비스 등