Smart Media Journal / Vol.15, No.7 / ISSN:2287-1322
https://dx.doi.org/10.30693/SMJ.2026.15.7.44

CXL 메모리 환경에서 GLnexus 기반 대형 작물 유전체 변이 분석 파이프라인의 메모리 배치 특성 분석

(Characterization of Memory Placement in a GLnexus-based Large Plant Genome Variant Analysis Pipeline on CXL Memory)

송영호*, 이현조**, 정기문***, 정현미***, 채철주**
(Youngho Song, Hyunjo Lee, Kimoon Jeong, Hyun Mi Jung, Chel-Joo Chae)

요 약

대형 작물 유전체 변이 분석에서는 다수의 샘플별 gVCF를 코호트 단위로 병합하는 과정에서 높은 메모리 사용량과 긴 실행 시간이 발생할 수 있다. 본 논문에서는 IPK Genebank Core 보리 데이터셋을 대상으로 GATK4 HaplotypeCaller, GLnexus 및 bcftools를 결합한 변이 분석 파이프라인을 구성하고, CXL 메모리 확장 환경에서 GLnexus 기반 gVCF 병합 단계의 메모리 배치 특성을 분석하였다. 실험은 4개, 8개, 16개, 24개 및 31개 샘플 부분집합을 대상으로 수행하였으며, DDR 단독, CXL 단독, DDR 우선, CXL 우선 및 DDR-CXL 가중 인터리빙 정책을 비교하였다. 실험 결과, 31개 샘플 조건에서 DDR 단독 및 CXL 단독 배치는 일부 염색체 병합 단계에서 실패하였으나, DDR 우선, CXL 우선 및 가중 인터리빙 정책은 전체 파이프라인을 완료하였다. 또한 성공한 정책에서는 동일한 샘플 수에 대해 최종 SNP 수가 동일하게 나타났다. 이러한 결과는 CXL 메모리가 로컬 DRAM의 단순 대체 자원이 아닌 DDR과 함께 사용하는 계층형 메모리 확장 자원으로 활용될 때 대규모 gVCF 병합 작업의 실행 안정성을 높일 수 있음을 보여준다.

■ 중심어 : 대형 작물 유전체 ; 결합 유전형 판별 ; GLnexus ; CXL 메모리 ; 메모리 배치

Abstract

Large plant genome variant analysis can require substantial memory capacity and long execution time when merging per-sample gVCFs at the cohort level. In this paper, we construct a variant analysis pipeline that combines GATK4 HaplotypeCaller, GLnexus, and bcftools using the IPK Genebank Core barley dataset, and analyze the memory placement characteristics of the GLnexus-based gVCF merging stage in a CXL memory expansion environment. Experiments were conducted with 4, 8, 16, 24, and 31 sample subsets under DDR-only, CXL-only, DDR-preferred, CXL-preferred, and DDR-CXL weighted interleaving policies. The results show that, in the 31-sample condition, DDR-only and CXL-only placements failed during some chromosome-level merging stages, whereas DDR-preferred, CXL-preferred, and weighted interleaving policies completed the entire pipeline. In addition, the final SNP counts were identical across successfully completed policies for the same sample size. These results indicate that CXL memory is more effective as a tiered memory expansion resource used together with local DDR, rather than as a simple replacement for local DRAM, for improving the execution stability of large-scale gVCF merging.

■ keywords : Large Plant Genome ; Joint Genotyping ; GLnexus ; CXL Memory ; Memory Placement

Ⅰ. 서 론

차세대 염기서열 분석 기술의 발전으로 다수의 개체를 대상으로 한 전장유전체 변이 분석

* 정회원, 전북대학교 컴퓨터공학과
** 정회원, 한국농수산대학교 교양학부
*** 정회원, 한국과학기술정보연구원 슈퍼컴퓨팅기술개발센터
이 논문은 2026년도 한국과학기술정보연구원(KISTI)의 기본사업으로 수행된 연구입니다.(과제번호:K26L3M2C2)

접수일자 : 2026년 07월 12일
게재확정일 : 2026년 07월 28일
교신저자 : 정현미, 채철주 e-mail : hmjung@kisti.re.kr, chae.chedjoo@gmail.com

(whole-genome variant analysis)이 생명정보학 연구에서 핵심적인 분석 절차로 활용되고 있다. 코호트 단위의 변이 분석에서는 개별 샘플에서 생성된 변이 호출 결과를 통합하여 집단 수준의 유전형을 판별하는 결합 유전형 판별(joint genotyping) 과정이 필요하다. 이러한 과정은 일반적으로 GATK HaplotypeCaller를 이용하여 샘플별 gVCF를 생성한 뒤, 코호트 단위로 병합하고 최종 VCF를 생성하는 방식이 널리 사용된다[1, 2].

그러나 코호트 규모가 증가할수록 gVCF 병합 단계에서 처리해야 하는 입력 파일 수와 중간 데이터 규모가 증가하며, 이에 따라 실행 시간, 저장 공간 및 메모리 사용량이 크게 증가한다. 이러한 문제는 인간 유전체보다 염색체 길이가 길고 반복 서열이 많은 대형 작물 유전체 분석에서 더욱 커질 수 있다. 특히 보리와 같은 대형 작물 유전체는 주요 염색체의 길이가 매우 길기 때문에 압축 VCF/gVCF 처리 과정에서 사용되는 인덱스 형식의 좌표 범위 제한이나 구간 분할 문제 등 추가적인 한계가 발생할 수 있다. 따라서 대형 작물 유전체 데이터에 대해 기존 변이 분석 파이프라인을 적용하기 위해서는 병합 단계의 확장성과 메모리 사용 특성을 분석하는 것이 중요하다.

GLnexus는 다수의 gVCF를 효율적으로 병합하기 위해 RocksDB 기반의 영속적 데이터베이스(persistent database)와 결합 변이 호출(joint variant calling) 알고리즘을 활용하는 도구이다[3, 4]. GLnexus는 대규모 코호트 변이 병합에 적합한 구조를 제공하지만, 다수의 gVCF 입력, RocksDB 기반 중간 데이터 관리, BCF/VCF 변환 및 후처리 과정에서 여전히 상당한 메모리 사용량을 요구한다. 따라서 GLnexus 기반 병합 단계는 대형 작물 유전체 분석 파이프라인에서 데이터 집약적 병목 구간이 될 수 있다.

한편, CXL(Compute Express Link)은 CPU와 메모리 확장 장치를 연결하기 위한 캐시 일관성 인터커넥트 기술로, 기존 로컬 DRAM의 용량 한계를 완화할 수 있는 계층형 메모리 확장 기술로 주목받고 있다[5]. CXL 메모리는 서버 시스템에서 추가적인 메모리 용량을 제공할 수 있지만, 로컬 DRAM과 비교하여 접근 지연시간과 대역폭 특성이 다를 수 있으므로, 단순히 메모리 용량을 확장하는 것만으로는 응용 성능을 보장하기 어렵다. 따라서 CXL 메모리 확장 환경에서는 워크로드의 메모리 접근 특성과 데이터 배치 방식에 따른 성능 변화를 분석할 필요성이 존재한다.

기존 연구에서는 GLnexus의 대규모 코호트 처리 성능과 정확도, 또는 CXL 메모리 기반 범용 HPC 및 AI 워크로드의 메모리 배치 특성이 각각 독립적으로 연구되어 왔다. 그러나 대형 작물 유전체 변이 분석과 같이 대규모 gVCF 병합 과정에서 높은 메모리 사용량이 요구되는 생명정보학 워크로드를 대상으로, CXL 메모리 배치 정책이 실행 안정성과 처리 특성에 미치는 영향을 분석한 연구는 아직 제한적이다. 따라서 본 연구의 목적은 CXL 메모리 자체의 성능을 단순히 평가하는 것이 아니라, GLnexus 기반 대형 작물 유전체 변이 분석 파이프라인에서 발생하는 메모리 병목을 DDR/CXL 계층형 메모리 배치로 완화할 수 있는지 분석하는 데 있다.

본 연구에서는 IPK Genebank Core 보리 데이터셋[6]을 대상으로 GATK4 HaplotypeCaller, GLnexus 및 bcftools를 결합한 대형 작물 유전체 변이 분석 파이프라인을 구성하고, CXL 메모리 확장 환경에서 GLnexus 기반 gVCF 병합 단계의 성능과 메모리 사용 특성을 분석한다. 이를 위해 4개, 8개, 16개, 24개 및 31개 샘플 부분집합을 구성하고, DDR 단독 배치, CXL 단독 배치, DDR 우선 배치, CXL 우선 배치 및 DDR-CXL 가중 인터리빙 정책을 적용하여 실행 시간, 메모리 사용량, 실행 성공 여부 및 최종 변이 호출 결과의 일관성을 비교하였다.

본 연구의 주요 기여는 다음과 같다. 첫째, IPK Genebank Core 보리 데이터셋을 대상으로

GATK4 HaplotypeCaller, GLnexus 및 bcftools를 결합한 대형 작물 유전체 변이 분석 파이프라인을 구성하였다. 둘째, GLnexus 기반 gVCF 병합 단계에서 샘플 수 증가에 따른 실행 시간과 메모리 사용량 변화를 분석하여, 대형 작물 유전체 분석에서 병합 단계가 주요 병목으로 작용함을 확인하였다. 셋째, 기존 GLnexus 연구가 대규모 코호트 병합의 정확도와 확장성을 중심으로 평가하고, 기존 CXL 연구가 범용 HPC 및 LLM 워크로드를 중심으로 메모리 배치 정책을 분석한 것과 달리, 본 연구는 대형 작물 유전체 gVCF 병합 파이프라인을 대상으로 DDR/CXL 메모리 배치 정책의 효과를 실험적으로 분석하였다. 이를 통해 CXL 메모리를 로컬 DDR의 단순 대체 자원으로 사용하는 것보다 DDR과 함께 사용하는 계층형 메모리 확장 자원으로 활용하는 것이 대규모 gVCF 병합 작업의 실행 안정성을 높일 수 있음을 보였다.

II. 관련 연구

1. GATK/GLnexus 기반 대규모 결합 유전형 판별

GATK[1, 2]의 HaplotypeCaller를 이용하여 샘플별 gVCF를 생성한 뒤, GenomicsDB-Import와 GenotypeGVCFs를 통해 코호트 단위의 유전형을 통합 판별하는 결합 유전형 판별 방식은 유전체 기반 코호트 변이 분석에서 널리 사용되는 표준 파이프라인이다. 그러나 코호트 규모가 증가할수록 gVCF 병합 및 유전형 판별 과정에서 실행 시간, 저장 공간, 메모리 사용량 등 계산 오버헤드가 증가하는 확장성 문제가 발생한다. 또한 대형 작물 게놈과 같이 개별 염색체 길이가 긴 참조 유전체를 사용하는 경우, 압축 VCF/gVCF에 사용되는 TBI 인덱스의 염색체 길이 제한으로 인해 기존 GATK 기반 병합 파이프라인 적용이 제한될 수 있다. 이러한 한계는 인간 유전체 연구에서 널리 사용·검증된 기존 파이프라인을 대형 작물 유전체 분석에 적용할 때 추가적인 기술적 제약으로 작용한다.

이러한 확장성 한계를 완화하기 위해 M. F. Lin et al.(2018)은 로그 구조 병합 트리(log-structured merge tree, LSM tree) 기반의 키-값 저장소인 RocksDB를 활용하여, 추가 샘플이 입력될 때 효율적으로 확장 가능한 영속적 데이터베이스와 결합 변이 호출 알고리즘을 결합한 GLnexus를 제안하였다[3]. 해당 연구는 50,000개 엑솜(exome) 데이터를 이용해 GLnexus의 성능을 검증하였고, DNAnexus 기반 대규모 클라우드 환경에서 240,000개 이상의 엑솜 및 22,000개 이상의 전장유전체(whole genome) 코호트에 적용 가능함을 보였다.

T. Yun et al.(2020)은 GATK HaplotypeCaller 대신 딥러닝 기반 변이 호출 도구인 DeepVariant를 사용하여 샘플별 gVCF를 생성하고, 이를 GLnexus로 병합하는 DeepVariant+GLnexus 파이프라인을 제안하였다[7]. 해당 연구는 다양한 코호트 규모, 시퀀싱 깊이(sequencing depth), 시퀀싱 방법에 대해 벤치마크 샘플의 정밀도(precision)와 재현율(recall), 부모-자녀 trio의 멘델 일치도(Mendelian consistency)를 기준으로 성능을 평가하였다. 실험 결과, DeepVariant 기반 gVCF의 전체 파일 크기는 GATK HaplotypeCaller 기반 gVCF의 약 1/7 수준이었으며, 최종 코호트 VCF 역시 GATK 파이프라인보다 26% 작은 용량을 보였다. 또한 1KGP 2,504개 샘플의 22번 염색체 gVCF를 코호트 VCF로 병합하는 실험에서, 단일 32-vCPU 가상 머신 기준 GLnexus는 0.84시간, GATK의 GenomicsDBImport 및 GenotypeGVCFs 조합은 6.83시간이 소요되어 GLnexus 기반 병합이 약 8배 빠른 실행 시간을 보였다. 이는 단일 샘플 변이 호출 도구와 gVCF 병합 도구의 조합에 따라 정확도와 계산 효율이 달라질 수 있음을 보인다. 해당 연구는 DeepVariant로 생성한 gVCF를 GLnexus로 병합한 경우를 중심으로 평가하였

으며, 본 연구는 GATK4 HaplotypeCaller로 생성한 gVCF를 GLnexus에 입력한다는 점에서 파이프라인 구성의 차이가 존재한다.

2. 대형 작물 유전체 데이터셋과 변이 분석

Z. Yuan et al.(2025)은 IPK 유전자은행의 보리 핵심 1000 집단에서 선별한 812개 보리 식물유전자원과 298개 유럽 우수 육종 소재를 포함하여 총 1,110개 유전형에 대한 전장유전체 염기서열 분석 기반 유전형 자료를 구축하였다[6]. 보리는 인간 유전체와 비교해 염색체 길이가 길고 반복 서열 비중이 높은 대형 작물 유전체에 해당하므로, 다수 샘플의 gVCF를 코호트 단위로 병합할 때 입력 데이터 규모와 중간 데이터 규모가 크게 증가할 수 있다. 따라서 해당 데이터셋은 대형 작물 유전체 분석 파이프라인에서 gVCF 병합 단계의 메모리 사용 특성과 확장성을 분석하기에 적합하다.

본 연구에서는 해당 공개 데이터셋을 생물학적 형질 분석의 대상으로 사용하는 것이 아닌 대형 작물 유전체 변이 분석 파이프라인의 메모리 집약적 특성을 평가하기 위한 입력 데이터로 활용한다. 특히 보리와 같은 대형 작물 유전체에서는 인덱싱 제약, gVCF 병합 비용, 메모리 사용량 증가가 동시에 발생할 수 있으므로, 코호트 단위 병합 단계의 성능 및 메모리 병목을 분석하는 것이 필요하다.

3. CXL 기반 메모리 확장과 데이터 집약형 워크로드

CXL은 CPU와 메모리 확장 장치 및 가속기를 연결하기 위한 캐시 일관성 인터커넥트(cache-coherent interconnect) 기술로, 기존 로컬 DRAM 중심의 메모리 용량 한계를 완화하고 서버 시스템에 계층형 메모리 구조를 제공할 수 있다[5]. 이러한 계층형 메모리에서 로컬 메모리와 확장 메모리 사이의 접근 지연시간 및 대역폭 차이로 인해 데이터 배치와 메모리 관리 정책이 주요한 연구로 수행되었다. 이성민 등은 CXL 기반 계층형 메모리 시스템에서 접근 빈도와 메모리 상태를 고려한 동적 페이지 마이그레이션 기법을 제안하였고[8], 박지원 등은 강화학습 기반의 적응형 메모리 관리 기법을 통해 계층형 메모리 환경에서 페이지 배치 정책을 최적화하고자 하였다[9]. 또한 김우철 등은 분산 메모리 시스템에서 hot-cold 데이터 분류를 활용한 데이터 관리 기법을 연구하였다[10]. Wang et al.(2025)은 CXL 메모리 확장 환경에서 우선 할당, 메모리 바인딩, 균등 페이지 단위 인터리빙 및 데이터 객체 단위 인터리빙 정책이 HPC 및 LLM 워크로드 성능에 미치는 영향을 분석하였다[11]. 특히 균등 인터리빙은 선택된 메모리 노드들 사이에 페이지를 라운드 로빈 방식으로 분산 배치하는 방식이며, 워크로드 내부 데이터 객체의 접근 특성에 따라 일부 객체만 인터리빙하는 데이터 객체 단위 인터리빙 정책을 제안하였다. 해당 연구는 CXL 메모리가 CPU에서 접근 가능한 별도의 메모리 계층으로 동작하지만, 로컬 DRAM에 비해 상대적으로 높은 지연시간과 낮은 대역폭 특성을 가질 수 있음을 보였다. 또한 DRAM과 CXL 메모리를 함께 사용할 때 페이지 단위 인터리빙 및 메모리 계층화 정책이 응용 성능에 미치는 영향을 분석하고, 데이터 객체의 접근 특성에 따라 메모리 배치 방식을 조정하는 데이터 객체 단위 인터리빙 정책을 제안하였다.

해당 연구는 CXL 기반 메모리 확장 환경에서 단순히 부족한 메모리를 CXL로 보완하는 것만으로는 충분하지 않으며, 워크로드의 메모리 접근 패턴과 데이터 객체의 사용 특성을 고려한 배치 전략이 필요함을 시사한다. 그러나 기존 CXL 메모리 연구는 범용 HPC 응용과 LLM 워크로드를 중심으로 수행되었으며, gVCF 병합과 RocksDB 기반 중간 데이터 관리를 포함하는 유전체 분석 파이프라인은 직접적으로 다루지 않았다. 또한 기존 GLnexus 연구는 대규모 코호트 변이 병합의 정확도와 확장성을 주로 인간 유

전체 또는 엑솜 데이터 중심으로 평가하였으나, CXL 메모리 확장 환경에서 GLnexus 병합 단계의 메모리 배치 특성은 분석하지 않았다. 따라서 본 연구는 대형 작물 유전체 gVCF 병합 파이프라인을 대상으로 GLnexus 기반 병합 단계의 메모리 병목을 특성화하고, DDR/CXL 메모리 배치 정책이 성능과 메모리 사용량에 미치는 영향을 실험적으로 분석한다는 점에서 기존 연구와 차별성을 가진다.

III. 연구 방법 및 실험 설계

1. 분석 대상 데이터 셋

본 연구에서는 IPK Genebank Core 보리 데이터 셋에서 선별한 31개 샘플을 대상으로 실험을 수행하였다. 각 샘플에 대해서는 GATK4 HaplotypeCaller를 이용하여 gVCF 형식의 변이 호출 결과를 생성하였으며, 이를 이후 GLnexus 기반 코호트 병합 단계의 입력으로 사용하였다. 참조 유전체는 MorexV3[12]를 사용하였고, 보리의 7개 주요 염색체 구간을 대상으로 염색체별 병합을 수행하였다.

본 연구의 목적은 전체 보리 코호트에 대한 생물학적 변이를 해석하는 것이 아닌 대형 작물 유전체 변이 분석 파이프라인에서 GLnexus 기반 병합 단계의 메모리 사용 특성과 CXL 메모리 배치 정책의 영향을 분석하는 것이다. 따라서 본 연구에서는 실험 서버의 로컬 DRAM/CXL 메모리 구성, 저장장치 용량, gVCF 전처리 비용을 고려하여 우선 확보 및 전처리가 완료된 31개 샘플을 대상으로 실험을 수행하였다. 또한 31개 샘플로부터 4개, 8개, 16개, 24개 및 31개 부분집합을 구성하여, 동일한 파이프라인에서 샘플 수 증가에 따른 실행 시간과 메모리 사용량의 변화를 단계적으로 분석하였다.

2. GATK4-GLnexus-bcftools 기반 분석 파이프라인

본 연구에서는 대형 작물 유전체 변이 분석을 위해 GATK4, GLnexus 및 bcftools를 결합한 분석 파이프라인을 구성하였다. 먼저 원시 리드(read) 데이터인 FASTQ 파일을 BWA-MEM2 [13]를 이용하여 참조 유전체에 정렬하고, 정렬된 BAM 파일에 대해 중복 리드 마킹을 수행하였다[14, 15]. 이후 GATK HaplotypeCaller를 이용하여 샘플별 gVCF를 생성하고, 생성된 gVCF 파일을 GLnexus[3]의 입력으로 사용하여 염색체 단위의 결합 변이 호출을 수행하였다. GLnexus의 염색체별 결과는 bcftools[16]를 이용하여 VCF 형식으로 변환 및 병합하였으며, 최종적으로 SNP 추출 및 품질 필터링을 수행하여 최종 SNP VCF를 생성하였다.

그림 1은 본 연구에서 사용한 GATK4-GLnexus-bcftools 기반 변이 분석 파이프라인을 나타낸다. GATK4 기반 단계에서는 FASTQ 입력 데이터로부터 정렬 BAM을 생성하고, 중복 리드 마킹 및 HaplotypeCaller를 통해 샘플별 gVCF를 생성한다. 이후 GLnexus를 이용하여 염색체별 결합 변이 호출을 수행하고, bcftools를 통해 BCF/VCF 변환, 염색체별 VCF 병합, SNP 추출 및 품질 필터링을 수행하여 최종 SNP VCF를 생성한다.

본 연구에서 GLnexus 병합 단계를 주요 분석 대상으로 선택한 이유는 다음과 같다. 첫째, 보리 MorexV3 참조 유전체의 주요 염색체는 길이가 매우 길기 때문에 염색체 전체를 하나의 구간으로 처리하는 GATK 기반 병합 파이프라인에서는 인덱싱 및 구간 관리 제약이 발생할 수 있다. 둘째, GLnexus는 다수의 gVCF를 RocksDB 기반 중간 데이터 구조로 관리하며 코호트 단위 결합 변이 호출을 수행하므로, 대규모 gVCF 병합에 적합한 구조를 제공한다. 셋째, GLnexus 병합 단계는 다수의 압축 gVCF 입력, RocksDB 캐시 및 버퍼, BCF/VCF 출력 데이터를 동시에 처리하기 때문에 메모리 사용량이 크고, CXL 메모리 배치 정책의 영향을 분석하기에 적합한 데이터 집약적 병목 구간이다. 따라서 본 연구에서

는 GATK HaplotypeCaller 이후 단계 중 GLnexus 기반 gVCF 병합과 bcftools 후처리 단계를 성능 분석 범위로 설정하였다.

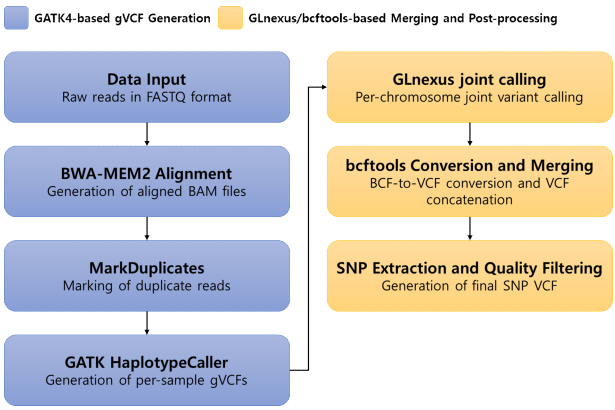

그림 1. GATK4-GLnexus-bcftools 기반 변이 분석 파이프라인

3. CXL 메모리 배치 정책

본 연구의 실험 서버는 로컬 DRAM이 연결된 NUMA node0과 CXL 메모리 확장 장치로 구성된 NUMA node1을 포함한다. node0은 CPU 코어가 연결된 로컬 DDR 메모리 노드이며, node1은 CPU 코어가 직접 연결되지 않은 CXL 메모리 노드로 인식된다. 본 연구에서는 GLnexus 병합 단계의 CPU 실행 위치를 node0으로 고정한 상태에서 메모리 할당 정책을 변경하여, CXL 메모리 배치가 성능과 메모리 사용 특성에 미치는 영향을 분석하였다.

본 연구에서는 로컬 DDR 메모리만 사용하는 DDR 단독 배치를 기준선으로 설정하고, CXL 메모리 확장 환경에서의 다양한 메모리 배치 정책을 비교하였다. 그림 2는 각 메모리 배치 정책에서 GLnexus 실행 중 작업 데이터가 DDR 및 CXL 메모리 노드에 배치되는 개념적 구조를 나타낸다. 그림의 각 박스는 실행 중 메모리에 적재되는 gVCF 입력 버퍼, GLnexus 작업 데이터, RocksDB 캐시 및 버퍼, BCF/VCF 출력 버퍼 등의 배치 개념을 의미한다.

비교한 메모리 배치 정책은 그림 2와 같이 (a) DDR 단독 배치, (b) CXL 단독 배치, (c) DDR 우선 배치, (d) CXL 우선 배치, (e) DDR-CXL 가중 인터리빙으로 구성하였다. DDR 단독 배치는 메모리 할당을 로컬 DDR 노드로 제한하는 방식으로, CXL 메모리를 사용하지 않는 기존 DDR 단독 실행 환경을 나타낸다. CXL 단독 배치는 메모리 할당을 CXL 노드로 제한하는 방식으로, CXL 메모리가 로컬 DDR 메모리를 대체할 수 있는지를 확인하기 위한 비교 조건이다. DDR 우선 배치는 로컬 DDR 메모리를 우선적으로 사용하고 필요 시 다른 메모리 노드를 함께 사용할 수 있는 정책이며, CXL 우선 배치는 CXL 메모리를 우선적으로 사용하고 필요 시 로컬 DDR 메모리를 함께 사용할 수 있는 정책이다. DDR-CXL 가중 인터리빙은 DDR과 CXL 메모리 간 페이지 할당 가중치를 조정하는 방식이다. 본 연구에서 설정한 가중 인터리빙 비율은 DDR 중심 배치, 균등 배치, CXL 중심 배치를 대표할 수 있도록 3:1, 1:1, 1:3으로 설정하였다. 3:1은 로컬 DDR 메모리의 낮은 접근 지연시간을 우선 활용하면서 CXL 메모리를 보조 확장 영역으로 사용하는 조건이며, 1:1은 DDR과 CXL 메모리에 페이지를 균등하게 분산하는 기준 조건이다. 1:3은 CXL 메모리 사용 비중을 높여 CXL 중심 배치가 GLnexus 병합 단계의 실행 안정성과 성능에 미치는 영향을 확인하기 위한 조건이다. 본 연구에서는 실험 시간이 긴 GLnexus 병합 작업의 특성을 고려하여 과도하게 많은 비율을 탐색하는 것이 아닌 세 가지 대표 가중치를 통해 CXL 사용 비중 변화에 따른 경향을 분석하였다. 여기서 가중 인터리빙 비율은 페이지 할당 가중치를 의미하며, 실제 관찰되는 DDR/CXL 메모리 사용량이 항상 동일한 비율로 나타나는 것은 아니다.

표 1. CXL 메모리 배치 정책

정책	설명
DDR 단독 배치	로컬 DRAM에만 메모리 할당
DDR 우선 배치	로컬 DRAM을 우선 사용
CXL 단독 배치	CXL에만 메모리 할당
CXL 우선 배치	CXL을 우선 사용
가중 인터리빙 3:1	DDR 중심 분산 배치
가중 인터리빙 1:1	DDR-CXL 균등 분산 배치
가중 인터리빙 1:3	CXL 중심 분산 배치

그림 2. CXL 메모리 배치 정책에 따른 GLnexus 작업 데이터의 메모리 배치 구조

4. 실험 환경 및 측정 지표

실험은 표 2와 같이 로컬 DRAM 256 GB와 CXL 메모리 256 GB로 구성된 서버 환경에서 수행하였다. 운영체제 커널은 Linux 6.17.0-35-generic이며, CXL 메모리 장치는 SMART Modular Technologies의 CXL 2.x memory device로 인식되었고, Linux 커널의 cxl_pci 드라이버를 통해 관리되었다. 메모리 배치 제어에는 weighted interleave 기능을 지원하는 numactl 2.0.19를 사용하였다. 분석 도구로는 GATK v4.6.2.0, GLnexus v1.4.1, bcftools 1.22 및 BWA-MEM2 2.2.1을 사용하였다. NUMA 구성에서 node0은 32개 CPU 코어와 약 256 GB 로컬 DRAM을 포함하며, node1은 CPU 코어가 없는 약 256 GB CXL 메모리 노드로 인식된다.

표 2. 실험 환경

항목	내용
CPU	Intel Xeon 6515P 3.8GHz
운영체제 커널	Linux 6.17.0-35
로컬 DRAM 메모리	256GB
CXL 메모리	256GB
CXL 장치	SMART Modular Technologies CXL 2.x memory device
NUMA 구성	node0: CPU 및 로컬 DRAM
	node1: CXL Memory
메모리 배치 제어 도구	numactl 2.0.19
변이 호출 도구	GATK v4.6.2.0
결합 변이 호출 도구	GLnexus v.1.4.1
후처리 도구	bcftools 1.22
정렬 도구	BWA-MEM2 2.2.1
참조 유전체	MorexV3
입력 데이터	GATK4 기반 샘플별 gVCF
분석 대상	IPK Genebank Core 보리 데이터셋

각 실험에서는 GLnexus 병합 단계와 bcftools 후처리 단계의 실행 시간, 최대 메모리 사용량, DDR 및 CXL 노드별 메모리 사용량, NUMA 접근 통계, 중간 데이터 크기 및 최종 VCF 통계를 수집하였다. 실행 시간과 최대 메모리 사용량은 프로세스 단위 성능 측정 도구를 이용하여 기록하였고, DDR 및 CXL 메모리 사용량은 NUMA 노드별 메모리 상태를 주기적으로 수집하여 산출하였다. 또한 NUMA 노드별 접근 특성을 확인하기 위해 시스템 수준의 NUMA 통계를 함께 기록하였다.

최종 변이 호출 결과의 일관성을 확인하기 위해 각 실험 조건에서 생성된 VCF에 대해 기본 변이 통계를 산출하였으며, SNP 수와 염색체별 변이 수를 비교하였다. 이를 통해 메모리 배치 정책 변화가 파이프라인의 성능에는 영향을 줄 수 있으나, 최종 변이 호출 결과의 내용에는 영향을 주지 않는지 확인하였다.

5. 실험 시나리오

본 연구의 실험 시나리오는 GLnexus 기반 대

형 작물 유전체 변이 분석 파이프라인에서 메모리 병목을 확인하고, CXL 메모리 배치 정책의 효과를 단계적으로 평가하기 위해 구성하였다. 표 3은 본 연구에서 수행한 네 가지 실험 시나리오를 정리한 것이다. 이 중 핵심 시나리오는 DDR/CXL 메모리 배치 정책 비교이며, 샘플 수 변화, 단계별 병목 분석 및 결과 일관성 검증은 메모리 배치 정책 비교 결과를 해석하기 위한 보조 실험으로 구성하였다.

첫째, 샘플 수 증가에 따른 GLnexus 병합 단계의 확장성을 분석하였다. 이를 위해 4개, 8개, 16개, 24개 및 31개의 샘플 부분집합을 구성하고, 동일한 분석 파이프라인을 적용하여 실행 시간과 메모리 사용량의 변화를 비교하였다. 이 실험은 샘플 수 증가에 따라 gVCF 병합 단계의 메모리 요구량이 증가하는지 확인하기 위한 기준 실험으로 사용하였다.

둘째, 동일한 입력 gVCF에 대해 메모리 배치 정책을 변경하여 GLnexus 병합 단계의 성능 변화를 분석하였다. 비교 대상은 DDR 단독 배치, CXL 단독 배치, DDR 우선 배치, CXL 우선 배치 및 DDR-CXL 가중 인터리빙 정책이다. DDR 단독 배치는 기존 DRAM-only 실행 환경을 나타내는 기준선으로 사용하였으며, CXL 단독 배치는 CXL 메모리가 로컬 DRAM을 대체할 수 있는지를 확인하기 위한 조건으로 설정하였다. DDR 우선 배치와 CXL 우선 배치는 각각 로컬 DRAM 또는 CXL 메모리를 우선적으로 사용하는 정책이며, DDR-CXL 가중 인터리빙은 DDR:CXL 페이지 할당 가중치를 3:1, 1:1, 1:3으로 조정하는 방식이다. 이를 통해 CXL 메모리 사용 방식과 사용 비중에 따른 실행 가능성, 실행 시간 및 메모리 사용 특성 변화를 비교하였다.

셋째, 전체 파이프라인을 GLnexus 병합 단계와 bcftools 후처리 단계로 구분하여 단계별 병목을 분석하였다. 이를 통해 gVCF 병합, 염색체별 VCF 병합, SNP 추출 및 품질 필터링 단계가 전체 실행 시간에서 차지하는 비중을 비교하였다. 이 실험은 CXL 메모리 배치 최적화의 주요 대상이 GLnexus 병합 단계인지, 또는 후속 bcftools 처리 단계인지 확인하기 위해 수행하였다.

마지막으로, 메모리 배치 정책 변화가 최종 변이 호출 결과에 미치는 영향을 확인하기 위해 결과 일관성 검증을 수행하였다. 각 실험 조건에서 생성된 최종 VCF에 대해 SNP 수와 기본 변이 통계를 비교하여, 메모리 배치 정책이 실행 성능에는 영향을 줄 수 있으나 최종 변이 결과의 구성에는 영향을 주지 않는지 확인하였다.

표 3. 실험 시나리오

실험	조건	측정 지표
샘플 수 변화	4, 8, 16, 24, 31	시간, 최대 메모리
메모리 배치 비교	DDR 단독, CXL 단독, DDR 우선, CXL 우선, 3:1, 1:1, 1:3	시간, DDR/CXL 사용량
단계별 병목 분석	GLnexus, concat, SNP 추출, filtering	단계별 시간, 메모리
결과 일관성 검증	정책별 최종 VCF	SNP 수, 염색체별 변이 수

Ⅳ. 실험 결과 및 분석

1. GATK 기반 병합 파이프라인의 적용 제약

본 연구에서 사용한 MorexV3 참조 유전체의 7개 주요 염색체 길이는 516,505,932bp에서 665,585,731bp 범위로 나타났다. 반면 GATK가 사용하는 TBI 인덱스는 개별 염색체에 대해 최대 229bp, 즉 536,870,912bp까지의 좌표를 지원한다[17, 18]. 본 연구의 실험 대상 염색체 7개 중 6개의 염색체가 이 한계를 초과하므로, 염색체 전체를 하나의 구간으로 처리하고 TBI 인덱스 방식으로 처리하는 것은 인덱싱 범위 초과로 인해 오류가 발생한다(표 4). 따라서 이러한 대형 작물 유전체 데이터베이스에서는 GLnexus의 CSI 인덱스를 사용하거나 염색체 구간을 더 작은 조각으로 분할하는 추가 처리가 필요하다.

본 연구에서는 이러한 인덱싱 및 구간 관리의 복잡성을 고려하여, GATK4 HaplotypeCaller로 생성한 샘플별 gVCF를 GLnexus에 입력하고, 염색체 단위의 결합 변이 호출을 수행하는 결합 파이프라인을 구성하였다. 이후 bcftools

기반 변환, 병합 및 필터링 단계에서는 CSI 인덱스를 사용하여 긴 염색체 좌표에 대응하였다.

표 4. MorexV3 주요 염색체 길이와 TBI 인덱스 한계 비교

염색체	길이(bp)	TBI 한계 초과 여부
LR890096.1	516,505,932	미초과
LR890097.1	665,585,731	초과
LR890098.1	621,516,506	초과
LR890099.1	610,333,535	초과
LR890100.1	588,218,686	초과
LR890101.1	561,794,515	초과
LR890102.1	632,540,561	초과

2. 샘플 수 증가에 따른 GLnexus 병합 성능

샘플 수 증가에 따른 GLnexus 병합 단계의 확장성을 분석하기 위해 4개, 8개, 16개, 24개 및 31개 샘플 부분집합에 대해 동일한 염색체별 병합 파이프라인을 수행하였다. 실험 결과, 샘플 수가 증가함에 따라 GLnexus 병합 단계의 실행 시간과 메모리 사용량은 전반적으로 증가하는 경향을 보였다.

4개 샘플 조건에서는 DDR 단독, CXL 단독, DDR 우선, CXL 우선 및 DDR-CXL 가중 인터리빙 정책이 모두 정상적으로 완료되었다. 4개 샘플 조건에서 DDR 단독 배치의 GLnexus 병합 시간은 약 64.1분으로 나타났으며, 최대 DDR 사용량은 약 85.9 GB, 최대 CXL 사용량은 0.1 GB 미만으로 관찰되었다(표 5). 이는 소규모 조건에서는 단일 DDR 메모리 노드만으로도 GLnexus 병합 작업을 안정적으로 처리할 수 있음을 보여준다.

반면 샘플 수가 증가할수록 단일 메모리 노드만 사용하는 정책의 한계가 나타났다. 4개, 8개, 16개 및 24개 샘플 조건에서는 DDR 단독 및 CXL 단독 배치가 모두 성공하였으나, 31개 샘플 조건에서는 DDR 단독 배치와 CXL 단독 배치가 일부 염색체 병합 단계에서 실패하였다.

이러한 실패는 전체 시스템의 물리적 메모리 용량이 절대적으로 부족했기 때문이 아니라, 메모리 할당이 단일 NUMA 메모리 노드로 제한되면서 GLnexus 병합 과정의 작업 데이터가 특정 메모리 노드에 집중되었기 때문으로 해석된다. GLnexus는 다수의 샘플별 gVCF를 동시에 입력받아 염색체 단위로 결합 변이 호출을 수행하며, 이 과정에서 gVCF 입력 버퍼, RocksDB 기반 중간 데이터, 캐시 및 BCF/VCF 출력 버퍼가 함께 사용된다. 샘플 수가 증가하면 이러한 작업 데이터의 규모가 함께 증가하므로, DDR 단독 또는 CXL 단독 배치에서는 해당 노드의 메모리 할당 여유 공간이 빠르게 감소하여 일부 병합 단계가 완료되지 못한 것으로 판단된다. 반면 DDR과 CXL을 함께 사용하는 우선 배치 및 가중 인터리빙 정책은 작업 데이터가 두 메모리 노드에 분산되거나 추가 메모리 노드를 활용할 수 있으므로, 단일 노드의 용량 한계를 완화하여 전체 병합 과정을 완료할 수 있었다.

이러한 실험 결과는 GLnexus 기반 gVCF 병합 단계가 샘플 수 증가에 따라 메모리 요구량이 증가하는 데이터 집약적 병목 구간임을 보인다. 특히 31개 샘플 조건에서 단일 메모리 노드만 사용하는 방식은 안정적인 실행을 보장하지 못하였으며, DDR과 CXL 메모리를 함께 사용하는 계층형 메모리 배치 전략이 필요함을 보인다.

3. CXL 메모리 배치 정책별 성능 비교

메모리 배치 정책별 성능을 비교하기 위해 DDR 단독, CXL 단독, DDR 우선, CXL 우선 및 DDR-CXL 가중 인터리빙 정책을 적용하였다. DDR 단독 배치는 CXL 메모리를 사용하지 않는

표 5. 샘플 수 및 메모리 배치 정책별 실행 성공 여부

샘플 수	DDR 단독	CXL 단독	DDR 우선	CXL 우선	가중 3:1	가중 1:1	가중 1:3
4	성공	성공	성공	성공	성공	성공	성공
8	성공	성공	성공	성공	성공	성공	성공
16	성공	성공	성공	성공	성공	성공	성공
24	성공	성공	성공	성공	성공	성공	성공
31	실패	실패	성공	성공	성공	성공	성공

기준선으로 설정하였으며, CXL 단독 배치는 CXL 메모리가 로컬 DDR을 대체할 수 있는지를 확인하기 위한 조건으로 사용하였다. DDR 우선 및 CXL 우선 배치는 각각 특정 메모리 노드를 우선적으로 사용하고, 필요 시 다른 메모리 노드를 함께 사용하는 정책이다. DDR-CXL 가중 인터리빙은 DDR:CXL 페이지 할당 가중치를 3:1, 1:1, 1:3으로 조정하는 방식으로 구성하였다.

실험 결과, 소규모 샘플 조건에서는 단독 배치와 우선 배치가 모두 정상적으로 실행되었다. 그러나 31개 샘플 조건에서는 표 6과 같이 DDR 단독 및 CXL 단독 배치가 일부 염색체 병합 단계에서 실패하였다. DDR 단독 배치의 최대 DDR 사용량은 약 250.3 GB로 로컬 DDR 노드의 용량 한계에 근접하였고, CXL 단독 배치의 최대 CXL 사용량은 약 256.0 GB로 CXL 노드의 용량 한계에 근접하였다. 이는 단일 메모리 노드에만 의존하는 방식이 대규모 gVCF 병합 작업에서 안정적이지 않을 수 있음을 보인다.

반면 DDR 우선, CXL 우선 및 DDR-CXL 가중 인터리빙 정책은 31개 샘플 조건에서 모두 전체 염색체 병합과 후처리 과정을 완료하였다. DDR 우선 배치에서는 최대 DDR 사용량이 약 250.1 GB, 최대 CXL 사용량이 약 139.7 GB로 나타났으며, CXL 우선 배치에서는 최대 DDR 사용량이 약 140.7 GB, 최대 CXL 사용량이 약 255.7 GB로 나타났다. 이는 우선 배치 정책이 특정 메모리 노드를 중심으로 사용하되, 필요 시 다른 메모리 노드를 함께 사용함으로써 단일 노드의 용량 한계를 완화할 수 있음을 의미한다.

가중 인터리빙 정책에서도 DDR:CXL 페이지 할당 가중치에 따라 두 메모리 노드의 사용 비중이 달라지는 경향이 관찰되었다. 31개 샘플 조건에서 가중 인터리빙 3:1은 DDR 중심의 메모리 사용 패턴을 보였고, 1:1은 DDR과 CXL 사용량이 비교적 균형 있게 분산되었으며, 1:3은 CXL 중심의 메모리 사용 패턴을 보였다. 세 가중 인터리빙 조건은 모두 31개 샘플 병합을 완료하였다.

처리 시간 측면에서는 성공적으로 완료된 정책들 간의 차이가 제한적이었다. 31개 샘플 조건에서 GLnexus 병합 시간은 DDR 우선 배치 123.7분, CXL 우선 배치 125.6분, 가중 인터리빙 3:1, 1:1, 1:3 조건에서 각각 124.5분, 126.9분, 125.5분으로 나타났으며, 최대 차이는 약 2.6% 수준이었다. DDR 사용 비중이 높은 정책에서 실행 시간이 근소하게 짧은 경향은 있었으나, GLnexus 병합 단계는 메모리 접근, RocksDB 기반 중간 데이터 관리, 압축 파일 처리 및 I/O가 함께 작용하는 복합적인 작업이므로, 본 실험 결과만으로 특정 요인에 의한 성능 차이를 단정하기는 어렵다. 따라서 본 연구에서 CXL 메모리 배치 정책의 효과는 실행 시간 단축보다 단일 메모리 노드의 용량 한계를 완화하여 파이프라인 실행 가능성을 높인 점에서 더 명확하게 나타났다.

따라서 CXL 메모리는 로컬 DDR의 완전한 대체 자원이 아니라 로컬 DDR과 함께 사용하는 계층형 메모리 확장 자원으로 활용될 때 대규모 GLnexus 병합 작업의 실행 가능성을 높일 수 있다. 특히 DDR 단독과 CXL 단독이 모두 실패한 31개 샘플 조건에서 우선 배치 및 가중 인터리빙 정책이 성공한 결과는, DDR-CXL 혼합 배치의 필요성을 보인다.

표 6. 31개 샘플 조건에서 메모리 배치 정책별 결과

정책	실행 결과	GLnexus 시간	bcftools 시간	최대 DDR 사용량	최대 CXL 사용량	최종 SNP 수
DDR 단독	실패	–	–	250.3 GB	0.2 GB	–
CXL 단독	실패	–	–	35.5 GB	256.0 GB	–
DDR 우선	성공	123.7분	38.3분	250.1 GB	139.7 GB	44,661,941
CXL 우선	성공	125.6분	38.4분	140.7 GB	255.7 GB	44,661,941
가중 인터리빙 3:1	성공	124.5분	38.5분	250.1 GB	158.9 GB	44,661,941
가중 인터리빙 1:1	성공	126.9분	38.4분	199.3 GB	192.4 GB	44,661,941
가중 인터리빙 1:3	성공	125.5분	38.4분	133.6 GB	255.7 GB	44,661,941

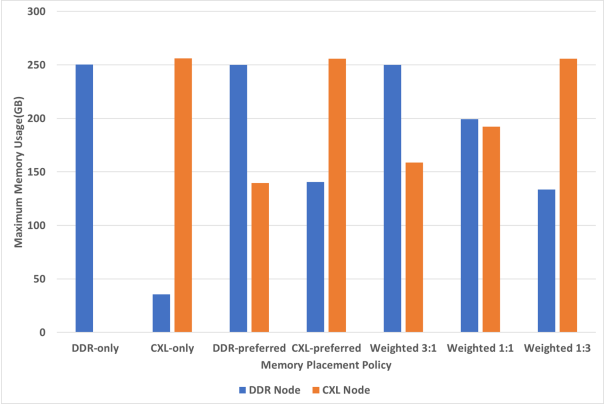

그림 3. 31개 샘플 조건에서 정책별 DDR/CXL 사용량

4. 파이프라인 단계별 병목 분석

전체 파이프라인의 병목 구간을 확인하기 위해 GLnexus 기반 염색체별 gVCF 병합 단계와 bcftools 기반 후처리 단계를 구분하여 실행 시간을 비교하였다. GLnexus 단계는 각 염색체에 대해 다수의 샘플별 gVCF를 동시에 입력받아 결합 변이 호출을 수행하므로, 전체 실행 시간의 대부분을 차지하였다.

이러한 결과는 GLnexus 단계가 단순한 파일 변환 작업이 아니라, 다수의 샘플별 gVCF를 동시에 해석하고 염색체 단위의 유전형 정보를 통합하는 계산 및 메모리 집약적 단계이기 때문으로 해석된다. 반면 bcftools 후처리 단계는 병합된 VCF/BCF 파일을 대상으로 변환, 병합, SNP 추출 및 필터링을 수행하므로, 샘플 수 증가에 따른 최종 변이 수와 파일 크기 증가의 영향을 주로 받는다.

또한 성공적으로 완료된 정책들을 기준으로 분석한 결과, 샘플 수가 증가함에 따라 GLnexus 병합 시간과 bcftools 후처리 시간이 모두 증가하였다. 4개 샘플 조건에서 평균 GLnexus 병합 시간은 약 65.0분, bcftools 후처리 시간은 약 3.8분으로 나타났으며, GLnexus 단계가 전체 측정 시간의 약 94.5%를 차지하였다. 31개 샘플 조건에서는 평균 GLnexus 병합 시간이 약 125.2분, bcftools 후처리 시간이 약 38.4분으로 증가하였고, GLnexus 단계의 비중은 약 76.5%로 나타났다.

bcftools 후처리 단계의 비중은 샘플 수가 증가함에 따라 함께 증가하였다. 이는 최종 VCF 크기와 변이 수가 증가하면서 VCF 병합, SNP 추출 및 품질 필터링 단계의 처리 비용도 증가하기 때문이다. 그러나 전체적으로는 GLnexus 병합 단계가 가장 큰 실행 시간 비중을 차지하였으며, 주요 병목은 단순한 VCF 후처리보다 코호트 단위 gVCF 병합 및 결합 변이 호출 단계에 있는 것으로 나타났다.

따라서 CXL 메모리 배치 정책의 성능 영향도 주로 GLnexus 병합 단계에서 나타난다. 이는 대형 작물 유전체 변이 분석 파이프라인에서 CXL 메모리 배치 최적화의 주요 대상이 GLnexus 기반 병합 단계임을 의미한다.

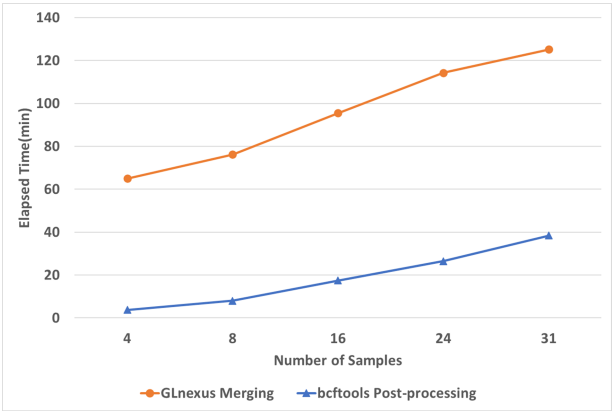

그림 4. 샘플 수 증가에 따른 단계별 평균 실행 시간

5. 최종 변이 호출 결과 검증

메모리 배치 정책이 최종 변이 호출 결과에 영향을 주는지 확인하기 위해, 각 정책에서 생성된 최종 SNP VCF에 대해 기본 변이 통계를 비교하였다. 비교 항목은 최종 SNP 수, 염색체별 변이 수 및 최종 VCF 크기이다[19]. 동일한 입력 gVCF와 동일한 GLnexus/bcftools 파이프라인을 사용한 경우, 메모리 배치 정책은 실행 시간과 메모리 사용 위치에는 영향을 줄 수 있지만 변이 호출 알고리즘의 입력과 조건을 변경하지 않는다.

실험 결과, 표 7과 같이 성공적으로 완료된 메

모리 배치 정책들에서는 동일한 샘플 수에 대해 최종 SNP 수가 모두 동일하게 나타났다. 4개 샘플 조건에서는 최종 SNP 수가 17,523,211개였고, 8개 샘플 조건에서는 24,389,840개, 16개 샘플 조건에서는 33,219,014개, 24개 샘플 조건에서는 37,582,317개, 31개 샘플 조건에서는 44,661,941개로 나타났다. 동일한 샘플 수 내에서는 DDR 단독, CXL 단독, DDR 우선, CXL 우선 및 가중 인터리빙 등 메모리 배치 정책이 달라져도 성공적으로 완료된 조건의 최종 SNP 수는 동일하였다.

이는 DDR/CXL 메모리 배치 정책이 프로세스의 메모리 할당 위치와 실행 가능성에는 영향을 주지만, GLnexus와 bcftools의 변이 호출 알고리즘, 입력 gVCF 및 필터링 조건 자체를 변경하지 않기 때문으로 해석된다. 따라서 동일한 입력 데이터와 동일한 분석 절차가 유지되는 경우, 계층형 메모리 배치 정책을 적용하더라도 최종 변이 호출 결과의 재현성은 유지될 수 있음을 의미한다.

표 7. 샘플 수별 최종 SNP 수

샘플 수	성공 정책 수	최종 SNP 수
4	7	17,523,211
8	7	24,389,840
16	7	33,219,014
24	7	37,582,317
31	5	44,661,941

6. 실험 결과 분석

본 연구에서는 CXL 메모리 배치 정책이 GLnexus 기반 대형 작물 유전체 변이 분석 파이프라인의 실행 안정성과 메모리 사용 특성에 미치는 영향을 분석하였다. 실험 결과, 단일 메모리 노드만 사용하는 DDR 단독 및 CXL 단독 배치는 31개 샘플 조건에서 일부 염색체 병합 단계를 완료하지 못하였다. 이는 대규모 gVCF 병합 과정에서 작업 데이터가 특정 메모리 노드에 집중되고, 해당 노드의 메모리 할당 여유 공간이 빠르게 감소했기 때문으로 해석된다. 반면 DDR과 CXL을 함께 사용하는 우선 배치 및 가중 인터리빙 정책은 동일한 조건에서 전체 염색체 병합과 후처리 과정을 완료하였다.

이러한 결과는 CXL 메모리가 로컬 DDR을 대체하기 위한 독립적인 메모리 자원이라기보다, 계층형 메모리 구조에서 부족한 메모리 용량을 보완하는 확장 자원으로 활용될 때 효과적임을 의미한다. 특히 성공적으로 완료된 정책들 사이의 실행 시간 차이는 제한적이었으나, 실행 성공 여부에서는 명확한 차이가 나타났다. 따라서 본 실험에서 CXL 메모리 배치의 주요 효과는 실행 시간의 큰 단축보다는 메모리 용량 병목을 완화하여 파이프라인 실행 안정성을 확보한 점에서 더 크게 나타났다.

또한 메모리 배치 정책이 실행 성공 여부와 메모리 사용 위치에는 영향을 주었지만, 동일한 샘플 수에서 성공적으로 완료된 조건의 최종 SNP 수에는 차이가 나타나지 않았다. 이는 DDR/CXL 계층형 메모리 구조를 사용하더라도 동일한 입력 데이터와 동일한 분석 절차가 유지되는 경우, 생물학적 분석 결과의 재현성을 유지하면서 실행 안정성을 향상시킬 수 있음을 보여준다.

따라서 향후 CXL 기반 생명정보학 시스템에서는 단순히 메모리 용량을 확장하는 것뿐 아니라, GLnexus 내부 데이터 구조와 접근 빈도를 고려한 데이터 중심 메모리 배치 기법을 함께 적용하는 것이 보다 효과적인 성능 향상 방안이 될 수 있다. 특히 반복적으로 접근되는 인덱스, 캐시 및 현재 처리 중인 작업 데이터는 로컬 DDR에 배치하고, 상대적으로 접근 빈도가 낮거나 대용량 보관 성격이 강한 데이터는 CXL 메모리에 배치하는 방식의 CXL-aware 메모리 배치 전략이 필요하다[20].

Ⅴ. 결론 및 향후 연구

본 연구에서는 대형 작물 유전체 변이 분석 파이프라인에서 발생하는 메모리 병목을 분석하기 위해 IPK Genebank Core 보리 데이터셋을 대상으로 GATK4 HaplotypeCaller, GLnexus 및 bcftools를 결합한 분석 파이프라인을 구성하였

다. GATK4 HaplotypeCaller를 이용하여 샘플별 gVCF를 생성하고, 이를 GLnexus 기반 결합 변이 호출 단계의 입력으로 사용하였으며, 이후 bcftools를 이용하여 VCF 변환, 병합, SNP 추출 및 품질 필터링을 수행하였다.

실험 결과, 보리 MorexV3 참조 유전체의 주요 염색체는 길이가 매우 크기 때문에 TBI 인덱스의 개별 염색체 좌표 지원 한계를 초과하는 경우가 확인되었다. 이에 따라 대형 작물 유전체 데이터에서는 염색체 전체를 하나의 구간으로 처리하는 기존 방식에 제약이 발생할 수 있으며, CSI 인덱스 사용 또는 구간 분할과 같은 추가 처리가 필요함을 확인하였다. 본 연구에서는 이러한 제약을 고려하여 GLnexus와 bcftools 기반의 염색체별 결합 변이 호출 및 후처리 파이프라인을 적용하였다.

또한 샘플 수를 4개, 8개, 16개, 24개 및 31개로 증가시키며 GLnexus 병합 단계의 성능과 메모리 사용 특성을 분석하였다. 실험 결과, 샘플 수가 증가함에 따라 GLnexus 병합 시간과 메모리 사용량이 증가하는 경향을 보였으며, 특히 31개 샘플 조건에서는 DDR 단독 배치와 CXL 단독 배치가 일부 염색체 병합 단계에서 실패하였다. 반면 DDR 우선 배치, CXL 우선 배치 및 DDR-CXL 가중 인터리빙 정책은 31개 샘플 조건에서도 전체 염색체 병합과 후처리 과정을 완료하였다. 이는 CXL 메모리가 로컬 DDR의 단순한 대체 자원이 아닌 DDR과 함께 사용하는 계층형 메모리 확장 자원으로 활용될 때 대규모 gVCF 병합 작업의 실행 가능성을 높일 수 있음을 보인다. 아울러 성공적으로 완료된 메모리 배치 정책들 사이의 실행 시간 차이는 제한적이었으며, 본 실험에서 CXL 메모리 배치의 주요 효과는 실행 시간의 큰 단축보다는 단일 메모리 노드의 용량 한계를 완화하여 파이프라인 실행 안정성을 확보한 점에서 나타났다.

단계별 실행 시간 분석에서는 GLnexus 기반 gVCF 병합 단계가 전체 파이프라인 실행 시간의 가장 큰 비중을 차지하는 것으로 나타났다. bcftools 기반 후처리 단계 역시 샘플 수와 최종 VCF 크기가 증가함에 따라 처리 시간이 증가하였으나, 전체적인 병목은 코호트 단위 gVCF 병합 및 결합 변이 호출 단계에 집중되었다. 따라서 대형 작물 유전체 변이 분석 파이프라인에서 CXL 메모리 배치 최적화의 주요 대상은 GLnexus 기반 병합 단계임을 확인하였다.

최종 변이 호출 결과 검증에서는 성공적으로 완료된 모든 메모리 배치 정책에서 동일한 샘플 수에 대해 최종 SNP 수가 동일하게 나타났다. 이는 메모리 배치 정책이 실행 시간과 메모리 사용 위치에는 영향을 줄 수 있지만, 동일한 입력 데이터와 동일한 분석 절차가 유지되는 경우 최종 변이 호출 결과의 구성에는 영향을 주지 않음을 의미한다.

본 연구의 결과는 CXL 메모리가 단순히 부족한 메모리를 보완하는 확장 장치가 아니라, 대규모 생명정보학 워크로드에서 로컬 DDR과 함께 계층형 메모리 구조를 구성할 때 실행 안정성을 향상시킬 수 있음을 보여준다. 특히 GLnexus 기반 대형 작물 유전체 변이 분석과 같이 메모리 집약적인 응용에서는 메모리 배치 정책 자체가 파이프라인의 실행 가능성에 직접적인 영향을 줄 수 있다. 이는 향후 CXL 기반 서버 환경에서 대규모 유전체 분석을 포함한 다양한 생명정보학 응용으로 확장될 수 있는 가능성을 제시한다.

향후 연구는 본 연구에서 수행한 프로세스 단위의 DDR/CXL 메모리 배치 정책 비교를 확장하여, GLnexus 내부 RocksDB 작업 데이터의 접근 특성을 고려한 CXL-aware 메모리 배치 기법을 연구하는 것이다. 본 연구에서는 DDR 사용 비중이 높은 정책에서 실행 시간이 근소하게 짧은 경향이 관찰되었으나, GLnexus 병합 단계는 메모리 접근, RocksDB 기반 중간 데이터 관리, 압축 gVCF/BCF 처리 및 파일 I/O가 함께 작용하는 복합적인 작업이므로, 각 요인이 처리 시간에 미치는 영향을 분리하여 분석하는 데에 한계가 있었다. 따라서 향후 연구에서는 NUMA 노

드별 local/remote 접근 비율, page fault, 메모리 대역폭, I/O 대기 시간, RocksDB cache 및 compaction 특성 등을 함께 수집하여 DDR/CXL 배치 정책별 처리 시간 차이의 원인을 정량적으로 분석하고자 한다. 또한 반복적으로 접근되는 인덱스, 캐시 및 현재 처리 중인 작업 데이터는 hot data로 분류하여 로컬 DDR 메모리에 배치하고, 상대적으로 접근 빈도가 낮거나 대용량 보관 성격이 강한 cold data는 CXL 메모리에 배치하는 계층형 메모리 전략을 설계할 계획이다. 이를 통해 단순한 프로세스 단위 메모리 배치를 넘어, GLnexus 내부 데이터 접근 특성에 기반한 실행 안정성 및 처리 시간 개선 가능성을 함께 검증하고자 한다.

REFERENCES

[1] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, and M.A. DePristo, “The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data,” *Genome Research*, vol. 20, no. 9, pp. 1297-1303, Sep. 2010.

[2] M.A. DePristo, E. Banks, R. Poplin, K.V. Garimella, J.R. Maguire, C. Hartl, A.A. Philippakis, G. del Angel, M.A. Rivas, M. Hanna, A. McKenna, T.J. Fennell, A.M. Kernytsky, A.Y. Sivachenko, K. Cibulskis, S.B. Gabriel, D. Altshuler, and M.J. Daly, “A framework for variation discovery and genotyping using next-generation DNA sequencing data,” *Nature Genetics*, vol. 43, no. 5, pp. 491-498, May 2011.

[3] M.F. Lin, I. Rodeh, P. Penn, G. Bai, B.L. Reid, P. Krasheninina, and A. Salerno, “GLnexus: joint variant calling for large cohort sequencing,” *bioRxiv*, 343970, 2018.

[4] S. Dong, A. Kryczka, Y. Jin, and M. Stumm, “RocksDB: Evolution of development priorities in a key-value store serving large-scale applications,” *ACM Transactions on Storage*, vol. 17, no. 4, Article 26, Oct. 2021.

[5] D. Das Sharma, R. Blankenship, and D.S. Berger, “An Introduction to the Compute Express Link (CXL) Interconnect,” *ACM Computing Surveys*, vol. 57, no. 1, Article 17, 2024.

[6] Z. Yuan, M. Rembe, M. Mascher, et al., “High-quality phenotypic and genotypic dataset of barley genebank core collection to unlock untapped genetic diversity,” *GigaScience*, vol. 14, giae121, 2025.

[7] T. Yun, H. Li, P.-C. Chang, M.F. Lin, A. Carroll, and C.Y. McLean, “Accurate, scalable cohort variant calls using DeepVariant and GLnexus,” *Bioinformatics*, vol. 36, no. 24, pp. 5582-5589, Dec. 2020.

[8] 이성민, 신지우, 정진만, “계층형 메모리 시스템을 위한 동적 페이지 마이그레이션 기법,” *스마트미디어저널*, 제14권, 제9호, 9-16쪽, 2025년 9월

[9] 박지원, 정진만, “Soft Actor-Critic 기반 계층형 메모리 시스템에서의 적응형 메모리 관리 기법,” *스마트미디어저널*, 제15권, 제3호, 46-53쪽, 2026년 3월

[10] 김우철, 민동희, 홍지만, “분산메모리시스템에서의 핫콜드 데이터 분류를 이용한 복합 백업 기법,” *스마트미디어저널*, 제4권, 제3호, 16-23쪽, 2015년 9월

[11] X. Wang, J. Liu, J. Wu, S. Yang, J. Ren, B. Shankar, and D. Li, “Performance Characterization of CXL Memory and Its Use Cases,” *Proc. of 2025 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1048-1061, Milan, Italy, Jun. 2025.

[12] M. Mascher, T. Wicker, J. Jenkins, et al., “Long-read sequence assembly: a technical evaluation in barley,” *The Plant Cell*, vol. 33, no. 6, pp. 1888-1906, Jun. 2021.

[13] M. Vasimuddin, S. Misra, H. Li, and S. Aluru, “Efficient Architecture-Aware Acceleration of BWA-MEM for Multicore Systems,” *Proc. of 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 314-324, Rio de Janeiro, Brazil, May 2019.

[14] G.A. Van der Auwera, M.O. Carneiro, C. Hartl, R. Poplin, G. del Angel, A. Levy-Moonshine, T. Jordan, K. Shakir, D. Roazen, J. Thibault, E. Banks, K.V. Garimella, D. Altshuler, S. Gabriel, and M.A. DePristo, “From FastQ data to high-confidence variant calls: The Genome Analysis Toolkit best practices pipeline,” *Current Protocols in Bioinformatics*, vol. 43, no. 1, pp. 1-43, 2013.

[15] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and 1000 Genome Project Data Processing Subgroup, “The Sequence Alignment/Map format and SAMtools,” *Bioinformatics*, vol. 25, no. 16, pp. 2078-2079, Aug. 2009.

[16] P. Danecek, J.K. Bonfield, J. Liddle, J. Marshall, V. Ohan, M.O. Pollard, A. Whitwham, T. Keane, S.A. McCarthy, R.M. Davies, and H. Li, "Twelve years of SAMtools and BCFtools," *GigaScience*, vol. 10, no. 2, giab008, Feb. 2021.
[17] HTSlib, "tabix(1) manual page," https://www.htslib.org/doc/tabix.html, Jul., 9, 2026.
[18] H. Li, "Tabix: fast retrieval of sequence features from generic TAB-delimited files," *Bioinformatics*, vol. 27, no. 5, pp. 718-719, Mar. 2011.
[19] P. Danecek, A. Auton, G. Abecasis, C.A. Albers, E. Banks, M.A. DePristo, R.E. Handsaker, G. Lunter, G.T. Marth, S.T. Sherry, G. McVean, R. Durbin, and 1000 Genomes Project Analysis Group, "The variant call format and VCFtools," *Bioinformatics*, vol. 27, no. 15, pp. 2156-2158, Aug. 2011.
[20] H. Al Maruf, H. Wang, A. Dhanotia, J. Weiner, N. Agarwal, P. Bhattacharya, C. Petersen, M. Chowdhury, S. Kanaujia, and P. Chauhan, "TPP: Transparent Page Placement for CXL-Enabled Tiered-Memory," *Proc. of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Vancouver, Canada, Mar. 2023.

저 자 소 개

송영호(정회원)
2014년 전북대학교 IT정보공학과 학사 졸업
2016년 전북대학교 컴퓨터공학과 석사 졸업
2026년 전북대학교 컴퓨터공학과 박사 졸업
<주관심분야 : 병렬컴퓨팅, 빅데이터, 딥러닝>

이현조(정회원)
2008년 전북대학교 컴퓨터공학과 석사 졸업
2014년 전북대학교 컴퓨터공학과 박사 졸업
2015년~2016년 한국과학기술정보연구원 선임연구원
2017년~현재 한국농수산대학교 연구원
<주관심분야 : 병렬 질의처리, 암호화 빅데이터, AI>

정기문(정회원)
2001년 전남대학교 전산통계학 석사 졸업
2009년 전남대학교 정보보호학 박사 졸업
2001년~2004년 한국정보보호진흥원 연구원
2004년~2005년 국가정보원 연구원
2005년~현재 한국과학기술정보연구원 슈퍼컴퓨팅기술개발센터 책임연구원
<주관심분야 : HPC 클라우드, 이기종 컴퓨팅, 클라우드 보안>

정현미(정회원)
2010년 한남대학교 컴퓨터공학과 석사 졸업
2014년 한남대학교 컴퓨터공학과 박사 졸업
2012년~현재 한국과학기술정보연구원 슈퍼컴퓨팅기술개발센터 선임연구원
<주관심분야 : HPC, 이기종컴퓨팅, 병렬컴퓨팅, 클라우드>

채철주(종신회원)
2009년 한남대학교 컴퓨터공학과 박사 졸업
2009년~2013년 한국전자통신연구원 선임연구원
2013년~2016년 한국과학기술정보연구원 선임연구원
2016년~현재 한국농수산대학교 교양학부 교수
<주관심분야 : 빅데이터, 인공지능, 디지털농업>