다분야 데이터 관리를 위한 버전 제어 애플리케이션 비교에 관한 연구

한성근, 전인호, 이정철, 최훈*
한국과학기술정보연구원, *충남대학교
{sghan, inojeon, jclee}@kisti.re.kr, *hc@cnu.ac.kr

A Study on the Comparison of Version Control Applications for Multidisciplinary Data Management

Sunggeun Han, Inho Jeon, Jeongchoel Lee, Hoon Choi*
Korea Institute of Science and Technology Information, *Chungnam National Univ.

요 약

인공지능 기술이 발전함에 따라 다양한 연구 분야에서 최신의 인공지능 기술을 적용하고 있으며, 다양한 데이터 유형 및 대량의 파일들이 사용되고 있다. 다분야 이종 데이터를 다루는 데이터 플랫폼에서는 데이터를 효과적으로 관리하기 위해 버전 관리가 필수적이다. 본 논문에서는 데이터 버전 제어 애플리케이션인 Git, Git-LFS, DVC를 사용하여 데이터 파일의 크기 및 파일 개수에 따른 Push 성능 및 Pull 성능을 비교하였다. 데이터 파일의 크기 또는 전체 파일의 개수에 따라 가장 효율적인 버전 관리 솔루션을 적용함으로써 다분야 데이터 플랫폼 운용을 효과적으로 수행할 수 있다.

Ⅰ. 서 론

최근 인공지능 기술이 급속도로 발전함에 따라 다양한 분야에서 해당 기술을 적용하고자 하는 시도가 계속되고 있다. 특히, 노 코드(No-code) 기반의 연구가 계속되면서 전문적인 인공지능 지식이 없어도 쉽게 연구 분야에 최신의 인공지능 기술을 적용할 수 있도록 다양한 플랫폼과 도구들이 개발되고 있다[1]. MLOps[2]를 기반으로 하는 데이터 플랫폼은 지속적인 소프트웨어 개발 접근 방식인 DevOps[3]를 기반으로 머신 러닝 시스템 개발에서 배포의 모든 단계를 파이프라인(Pipeline)으로 구성하고 라이프사이클(Lifecycle)로 정의하여 사용자의 일련의 작업 과정을 손쉽게 구축하도록 하고 있다. 파이프라인의 각 단계별 작업에서는 작업 실행을 위한 실행 환경(running environment), 입력 및 출력 데이터, 실행 코드(program code) 등이 사용된다. 이러한 일련의 작업이 한 번만으로 끝난다면 관련된 리소스를 데이터베이스 또는 스토리지에 저장하는 것으로 끝나겠지만, 소프트웨어 구축-배포-서비스-재구축 등과 같이 순환적 라이프사이클을 지원한다면 각 단계마다 이루어지는 모든 활동이 모니터링 및 관리되어야 한다. 특히, 단계별 작업 실행과 관련된 리소스들은 버전 관리를 통해 특정 시점에서 소프트웨어가 실행될 수 있도록 소프트웨어 재현성(reproducibility)을 지원해야 한다. 따라서, 본 논문에서는 데이터 기반의 다양한 소프트웨어 플랫폼에서 라이프사이클을 지원하기 위해 작업의 각 단계마다 필요한 프로그램 코드, 입력 및 출력 데이터, 기타 리소스 등을 효율적으로 관리하기 위한 데이터 버전 도구를 비교하고 사용자가 구축하고자 하는 데이터 플랫폼의 특성에 적합한 버전 도구를 선택할 수 있도록 한다.

Ⅱ. 실험 환경

□ 시스템 환경

데이터 버전 관리 테스트를 위해 클라이언트-서버 구성을 사용한다. 클라이언트 및 서버 노드는 KI-CLOUD[4] 상에 가상 머신(Virtual Machine)으로 각각 1대씩 구축하였다. 서버에는 데이터 리포지터리(repository)를 구축하고, 클라이언트에서는 버전 관리 애플리케이션을 사용하여 데이터를 서버에 전송(Push)하거나, 서버에서 데이터를 다운로드(Pull)한다. <표 1>은 클라이언트 노드와 서버 노드의 사양을 나타낸다.

<표 1> 클라이언트 및 서버 노드 사양

구분	클라이언트 노드	서버 노드
CPU	AMD 3.0MHz 8 Cores	AMD 3.0MHz 8 Cores
MEMORY	16GB	16GB
HDD	500GB	500GB

□ 비교 대상 애플리케이션

- Git: 파일의 변경사항을 추적하고 사용자들 간 파일 작업을 조정하기 위한 무료 오픈 소스 버전 관리 시스템(Version Control System)이다[5]. 또한, DesOps의 소스 코드 관리를 위해 사용되는 도구이기도 하다. 사용자는 중요한 변경을 수행할 때마다 새 파일을 만드는 대신 Git을 사용하여 파일의 현재 상태를 저장할 수 있으며, 필요에 따라 각 버전의 파일을 검색 및 다운로드하여 사용할 수 있다.
- Git-LFS (Git Large File Storage): Git의 확장 버전으로 대용량 파일에 대한 효율적인 버전 관리를 위해 만들어졌다[6]. 대용량 파일을 정기적으로 수정할 경우, Git-LFS에서는 cloning(복제) 이나 fetching(가져오기) 과정이 아니라 checkout 과정에서만 대용량 파일을 다운로드함으로 초기 복제하는 시간을 단축한다.
- DVC (Data Version Control): 머신 러닝(ML) 프로젝트를 공유하고 재현할 수 있도록 구축되었으며, 대규모 데이터 세트, ML 모델 및 코드에 대한 버전 관리를 지원한다[7]. 크기가 작은 파일은 Git으로 처리하고, 대용량 파일은 별도의 태그를 사용하여 관리한다. 이 원화된 관리를 통해 필요할 때만 대용량 파일을 다운로드함으로써 버전 관리를 효율적으로 수행한다.

Ⅲ. 실험 결과 및 분석

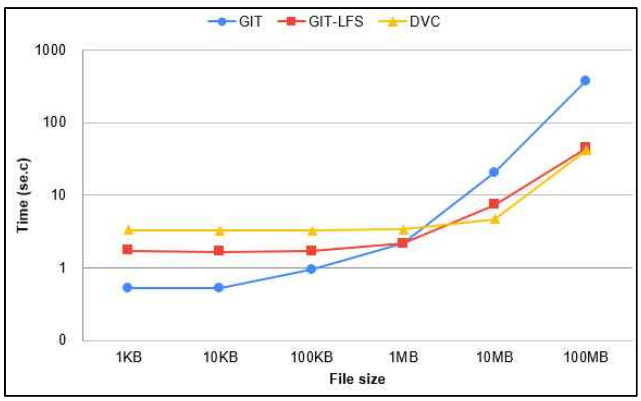
다분야 데이터 분석을 지원하는 데이터 플랫폼을 가정하고 파일 크기에 따른 성능 및 파일 개수에 따른 성능을 비교한다. 성능은 각 애플리케이션에서 제공하는 명령어에 소요되는 시간을 비교하였으며 크게 Push 시간(서버로 업로드하는 시간)과 Pull 시간(서버에서 다운로드하는 시간)으로 측정하였다. <표 2>는 명령어에 따른 Push 시간 및 Pull 시간을 계산하는 방법을 나타낸다. 예를 들면, DVC Push 시간은 데이터 업로드를 위해서 (dvc-add, git-add, git-commit, git-push, dvc-push)와 같이 5개의 명령이 필요하며, 5개 명령이 소요된 시간을 합산하여 계산한다.

<표 2> 성능 측정 방법

성능 / 비교 대상	Push 명령어	Pull 명령어
Git	Add, Commit, Push	Pull
Git-LFS	Add, Commit, Push	Pull
DVC	dvc-add, git-add, git-commit, git-push, dvc-push	git-pull dvc-pull

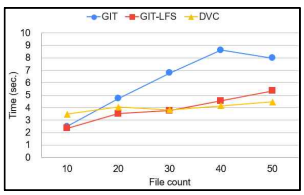
□ 파일 크기에 따른 성능 비교

다양한 크기를 가진 파일들에 대한 성능을 비교하기 위해 파일 하나의 크기를 1KB ~ 100MB까지로 하였으며, 각각 10개의 파일을 대상으로 버전 관리에 필요한 시간을 비교하였다. [그림 1]은 파일 크기에 따른 Push 시간 성능을 나타낸다. 1MB 이하의 적은 용량의 파일에 대해서는 Git이 우수하며, 파일 크기가 커질수록 Git-LFS와 DVC가 더 좋은 성능을 나타낸다.

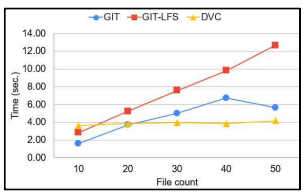


[그림 1] 파일 크기에 따른 Push 성능 비교

□ 파일 개수에 따른 성능 비교



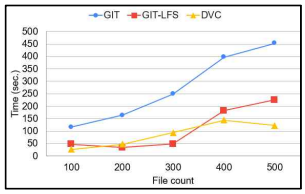
[그림 2] 1MB Push 성능



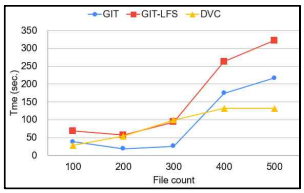
[그림 3] 1MB Pull 성능

대용량 파일의 성능을 비교하기 위해 파일 하나의 크기를 1MB, 10MB로 하여 파일 개수를 증가시키면서 성능을 비교하였다. [그림 2]와 [그림 3]은 1MB 크기의 파일을 10개에서 50개까지 증가시키면서 Push 성능과 Pull 성능 비교를 나타낸다. 파일의 개수가 30개 이하에서는 Git-LFS Push 성능이 가장 좋게 나타난다. 그러나, Git-LFS에서는 파일 개수가 증가할수록 Pull 성능이 급격하게 저하되는 문제가 있다.

[그림 4]와 [그림 5]는 10MB 크기의 파일을 100개에서 500개까지 증가시키면서 비교한 성능을 나타낸다. 파일 개수가 증가할수록 DVC가 Push 성능 및 Pull 성능 모두 우수한 것으로 나타난다. Git은 파일 개수가 증가할수록 Push와 Pull 성능 모두 급격히 나빠지며, Git-LFS는 Pull 성능이 급격히 나빠지는 문제를 가지고 있다.



[그림 4] 10MB Push 성능



[그림 5] 10MB Pull 성능

Ⅳ. 결론

융합연구를 위한 인공지능 기술 활용과 같은 다분야 대용량 이종 데이터 기반의 소프트웨어 라이프사이클을 지원하는 데이터 플랫폼을 구축할 경우, 데이터 버전 관리는 필수적이며 대용량 데이터에 대해서도 버전 제어 시스템(Version Control System)이 효율적으로 작동해야한다. 본 논문에서는 데이터 버전 관리 애플리케이션인 Git, Git-LFS, DVC를 사용하여 데이터 파일의 크기 및 파일 개수에 따라 파일을 업로드하는 Push 성능과 파일을 다운로드하는 Pull 성능을 비교하였다. 1MB 이하의 작은 크기 데이터 파일에 대해서는 Git 프로그램 성능이 가장 우수하였고, 파일 용량과 파일 개수가 증가할수록 DVC 프로그램이 가장 우수한 성능을 나타내었다. Git-LFS는 Push 성능은 우수하지만, 파일 개수가 증가할수록 Pull 성능이 급격히 저하되는 문제가 있었다. 본 성능 비교 결과를 통해 플랫폼에서 다루는 데이터 특성에 알맞은 버전 프로그램을 적용하여 효율적으로 사용할 수 있으리라 본다.

ACKNOWLEDGMENT

본 논문은 한국과학기술정보연구원에서 정부(과학기술정보통신부)의 지원(No.NRF-2022M3C1A6090416, 디지털융합R&D플랫폼 연구개발 및 서비스)로 수행된 연구임.

참 고 문 헌

[1] C. Richardson and J. R. Rymer. The forrester wave: Low-code development platforms, q2 2016. tech. rep. Forrester Research, 2016.

[2] Danilo Sato, Arif Wilder, and Christoph Windheuser. Continuous delivery for machine learning, Sept 2019.

[3] P Debois. Devops: A software revolution in the making. Cutter IT Journal, 24(8), 2011.

[4] Cho, Hyeyoung, et al. "Design and Development of KI Cloud Platform for High Performance Computing." Proceedings of the Korea Information Processing Society Conference. Korea Information Processing Society, 2020.

[5] Git, https://github.com/

[6] Git LFS, https://git-lfs.github.com/

[7] DVC, https://dvc.org/