NVMe-oF 를 이용한 Single-Machine-Based 그래프 엔진의 성능 측정

조익현 [1], 장명환 [2], 김상욱 [3]
[1] 한양대학교 컴퓨터·소프트웨어학과 석사과정
[2] 한양대학교 컴퓨터·소프트웨어학과 박사과정
[3] 한양대학교 컴퓨터·소프트웨어학과 교수

childyouth@hanyang.ac.kr, sugichiin@hanyang.ac.kr, wook@hanyang.ac.kr

Performance Evaluation of Single-Machine-Based Graph Engine using NVMe-oF

Ikhyeon Jo, Myung-Hwan Jang, Sang-Wook Kim*
Department of Computer and Software, Hanyang University

요 약

Single-machine-based 그래프 엔진은 단일 머신을 이용해 고성능의 그래프 분석을 가능하게 하지만 distributed-system-based 그래프 엔진보다 확장성이 낮다. 본 논문은 single-machine-based 그래프 엔진 중 state-of-the-art 인 RealGraph 에 NVMe-oF 기술을 이용한 고성능 원격 스토리지를 연결해 성능을 확인했다. 실험으로 우리는 고성능 원격 스토리지를 이용한 single-machine-based 그래프 엔진의 확장가능성이 있음을 확인하고 향후 연구에서 고성능 원격 스토리지를 사용할 경우 구조개선이 필요함을 제시한다.

1. 서론

그래프 분석은 객체 간의 관계성을 분석해 유용한 지식을 이끌어내는 기술로 다양한 분야에서 사용되고 있다. 특히 소셜 네트워크를 비롯한 실 세계 네트워크들의 크기가 폭발적으로 커져감에 따라 이러한 대규모 네트워크로부터 유용한 지식을 얻고자 다양한 그래프 알고리즘들이 사용되어 왔으며 [1, 2, 3, 4], 이러한 그래프 알고리즘들을 빠르게 수행하기 위한 수요가 증가하고 있다.

이러한 수요를 충족시키기 위해 빠르고 효율적으로 다양한 그래프 분석 알고리즘을 수행하기 위한 어플리케이션으로서, 그래프 엔진은 최근까지 single-machine-based 및 distributed-system-based 방식 등으로 다양한 연구가 진행되어 왔다 [5, 6, 7, 8, 9]. 특히 디스크 기반의 single-machine-based 방식은 많은 머신을 클러스터로 연결해 분석하는 distributed-system-based 그래프 엔진 [8, 9] 의 성능에 크게 뒤지지 않는 성능을 보여주어 왔다.

디스크 기반 single-machine-based 그래프 엔진 [5, 6, 7] 은 전체 그래프를 SSD 등의 외부 저장장치에 저장하고 그래프 처리에 필요한 일부를 반복적으로 메인 메모리로 적재하여 처리한다. 이를 통해 메인 메모리 용량을 초과하는 대규모 그래프도 처리할 수 있다. 그러나 여전히 단일 머신이 장착할 수 있는 디스크의 총 용량이라는 크기 제약을 벗어나지 못했다.

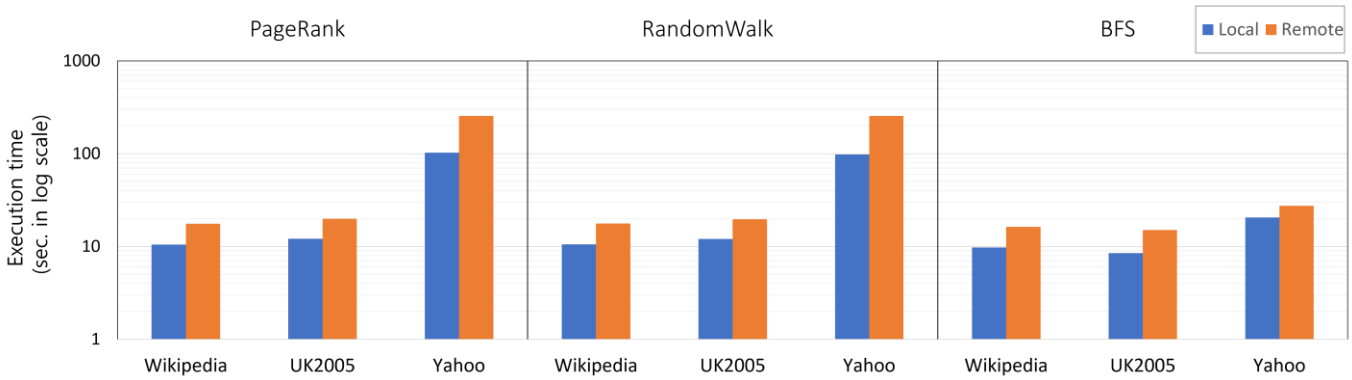
한편, PCIe 인터페이스를 활용하는 NVMe (Non-Volatile Memory Express) SSD 는 기존 SATA SSD 와 비교하여 수 배 빠른 IO 속도와 함께 다양한 기술을 제공하고 있다. 이러한 NVMe SSD 를 네트워크를 통해 연결하는 기술인 NVMe-oF (NVMe over Fabric)는 네트워크로 연결된 NVMe SSD 들을 로컬 저장장치처럼 사용할 수 있게 해줌으로써 싱글 머신에 한정된 외부 저장장치라는 한계를 벗어날 수 있다.

본 논문에서는 기존의 디스크 기반 single-machine-based 그래프 엔진의 state-of-the-art 중 하나인 RealGraph [5] 에 NVMe-oF 기술을 이용한 원격 저장장치로부터 네트워크를 통해 그래프를 전송 받아 처리할 수 있도록 하는 그래프 엔진을 개발하였다. 그리고 실험을 통해 개발한 시스템의 성능을 측정하고 결과를 분석한다.

2. 관련 연구

2.1 RealGraph

State-of-the-art 그래프 엔진 중 하나인 RealGraph 는



(그림 1) RealGraph^{remote} 모델과 RealGraph^{local} 모델의 수행시간 비교

단일 머신에서 동작하는 디스크 기반 그래프 엔진으로, 효율적인 그래프 분석 알고리즘 수행을 위해 storage, buffer, object, thread 총 4 개 계층으로 나뉘어 디스크에 저장된 그래프 데이터를 처리하게 된다.

RealGraph 는 실 세계 그래프가 power-law distribution 을 따른다는 점 [10] 을 이용하여 데이터 레이아웃을 구성한다. BFS 순회 순서대로 그래프의 정점 id 를 새로 할당하고 1MB 크기의 블록에 저장시킨다. 이러한 레이아웃은 디스크의 sequential read 확률과 메모리 캐싱 확률을 높여준다. RealGraph 의 데이터 레이아웃은 네트워크를 통한 연결에서도 효과적이다. 1MB 의 고정된 IO 단위를 가졌기 때문에 네트워크를 통한 데이터 전송량의 예측이 쉽고 효율적인 데이터 전송이 가능하다. 또한 원격 스토리지에서도 sequential read 확률을 올려줄 수 있다. RealGraph 는 각 블록을 쓰레드에 할당시켜 블록에 저장된 정점의 인접리스트들을 순차적으로 처리한다.

2.2 PoseidonOS

PoseidonOS 는 삼성에서 개발한 고성능의 경량 오픈소스 스토리지 OS 로, NVMe-oF 기술을 이용하여 네트워크를 통해 스토리지 서버를 제공할 수 있다. 고성능 스토리지 어플리케이션 개발을 위한 오픈소스 라이브러리인 SPDK(Storage Perfor-mance Development Kit) [11] 를 기반으로 동작한다.

고성능 스토리지 OS 달성을 위해 PoseidonOS 는 Userspace IO 로 동작하여 커널 경유로 인한 오버헤드를 최소화하고 polling 을 통해 IO 를 체크하여 인터럽트를 제거하는 등의 최적화를 적용시켰다. 여러 디스크를 관리하기 위해 RAID, logical volume 등의 유연한 블록 디바이스 관리가 가능하며 해당 장치들을 NVMe-oF 기술을 통해 네트워크로 노출시킬 수 있다. 스토리지 서버로서 요구하는 내결함성과 복구성을 갖춰 최고의 성능을 보이고 있다.

3. RealGraph^{remote}

우리는 기존 RealGraph 를 PoseidonOS 를 이용한 원격 스토리지와 연결함으로써 NVMe-oF 기반 원격 스토리지를 사용할 수 있게 하였다. 이를 위해 RealGraph 에 SPDK 기능들을 지원하도록 개발하고 PoseidonOS 를 네트워크에 노출시켜 통신을 수행하였다.

구체적으로, 우리는 PoseidonOS 에서 하나 이상의 NVMe SSD 들을 묶어 logical volume 을 구성하고 네트워크에 노출시킨다. RealGraph 는 노출된 volume 에 네트워크를 통해 연결하여 IO 를 요청한다. RealGraph 는 요청한 IO 에 대한 결과를 callback 이 아닌 polling 을 통해 확인하여 인터럽트를 최소화할 수 있도록 한다.

4. 실험

본 장에서는 RealGraph 가 설치된 프로세싱 서버와 PoseidonOS 가 설치된 스토리지 서버가 1 대 1 로 연결된 상황에서 RealGraph 의 성능을 local NVMe SSD 를 사용했을 때와 비교하여 성능을 측정하고 확장 가능성을 관찰하였다. 이를 위해 우리는 Xeon Gold 6336Y CPU 2 개와 SSD 32 개가 장착된 Inspur 서버 2 대를 각각 RealGraph 와 PoseidonOS 의 수행에 사용했다.

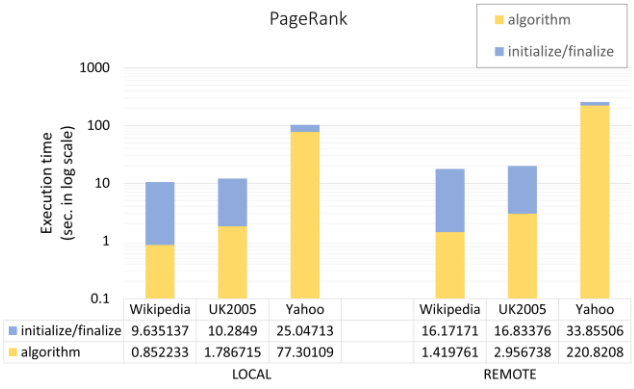
실험에는 Intel 의 SPDK (v23.09)와 PoseidonOS (v1.0.0)를 사용했다. PoseidonOS 측면에서는 총 32 개의 NVMe SSD 를 RAID0 로 묶은 단일 array 에서 logical volume 을 생성하였다. 생성된 logical volume 을 TCP transport 를 사용한 subsystem 에 등록시켜서 네트워크에 노출하였다. PoseidonOS 로 구성한 스토리지 서버와 연결한 모델 (RealGraph^{remote})과 비교하기 위해 local NVMe SSD 를 사용하는 모델 (RealGraph^{local})은 SPDK 를 이용하여 NVMe SSD 와 연결하였다. 실험에는 3 개의 실 세계 그래프 데이터 [5] <표 1> 에서 총 세 가지 알고리즘 – 각각 PageRank(PR), Random-Walk(RWR), BFS [1, 2, 3] 를 수행했다. RealGraph 수행 파라미터는 CPU 쓰레드를 32, 메모리 제한을 16GB,

queue depth 를 32 로 설정하였다. 두 Inspur 서버는 100Gbit Ethernet 으로 연결하였다.

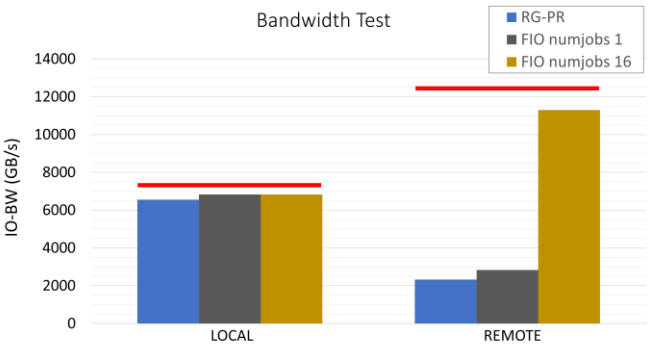
<표 1> 실 세계 그래프 데이터

데이터셋	Wikipedia	UK2005	Yahoo
# 정점	12M	39M	1.4B
# 간선	370M	930M	6.6B
저장 크기	5.7GB	16GB	114GB

(그림 1)은 RealGraph^{remote} 모델과 RealGraph^{local} 모델을 비교한 실험 결과로 알고리즘들의 데이터셋 별 수행 시간을 log scale 로 나타낸 그래프이다. 스토리지 서버에 연결함으로서 RealGraph^{remote} 모델이 32 배의 확장성을 얻었지만 전체적으로 수행시간이 늘어난 모습을 관찰할 수 있다. RealGraph^{remote} 모델은 최소 1.33 배에서 최대 2.56 배 까지 수행시간이 늘어난 것을 확인할 수 있었다. 우리는 RealGraph^{remote} 모델이 알고리즘 수행 뿐만 아니라 초기화 및 마무리 과정에서도 수행시간이 늘어난 것을 확인할 수 있었다. (그림 2)는 PageRank 의 initialize/finalize 와 algorithm 수행을 구분해 수행시간을 log scale 로 나타낸 그래프이다. 스토리지 서버와의 통신시간이 포함되는 algorithm 수행시간이 늘어남과 동시에 스토리지 접속 비용에 속하는 initialize/finalize 영역에서도 수행시간이 늘어났다. 그러나 데이터셋의 크기가 커짐에 따라 수행시간이 늘어날수록 initialize/finalize 영역의 수행시간보다 algorithm 의 수행시간 상승폭이 더 크게 증가하는 모습을 확인할 수 있다.



(그림 2) PageRank 의 algorithm 수행시간 및 initialize/finalize 수행시간 비교



(그림 3) FIO 및 RealGraph 의 IO Bandwidth 비교

두 모델 간 algorithm 수행시간의 차이가 큰 점에 있어 스토리지 대역폭 및 네트워크 대역폭을 완전히 사용하고 있는지 실험했다. (그림 3)은 오픈소스 IO 테스트 도구 중 하나인 FIO 를 SPDK 에서 제공하는 FIO 엔진을 이용했으며 numjobs 파라미터를 통해 IO 제출 thread 수를 조절해 local NVMe 와 스토리지 서버의 대역폭을 측정했다. FIO 대역폭은 RealGraph^{remote} 모델과 RealGraph^{local} 모델의 Page-Rank(Yahoo 데이터셋) 대역폭과 함께 비교했다. (그림 3)의 빨간 선은 각 IO 대상 별 최대 대역폭을 의미한다. Local NVMe SSD 를 사용한 경우 모든 상황에서 최대 대역인 7GB/s 에 근접한 결과를 보였다. 그러나 스토리지 서버를 사용한 경우 IO 제출 thread 의 개수가 1 개 일 때의 대역폭은 2.8GB/s 로 IO 제출 thread 의 개수가 16 개일 때의 대역폭인 11GB/s 와 큰 차이를 보였다. 이 때 RealGraph^{remote} 모델 역시 2.3GB/s 의 대역폭을 보였으며 이는 RealGraph^{remote} 모델의 IO 제출 thread 를 늘린다면 IO 대역폭 또한 늘어날 것이라고 생각할 수 있다.

우리는 두 실험을 통해 single-machine-based 그래프 엔진의 확장성 한계를 뛰어넘기 위한 방법으로 스토리지 서버에 연결하는 것이 유의미하며 단일 NVMe 대역폭에 비해 네트워크 대역폭이 큰 경우 확장성 뿐만 아니라 속도향상 가능성 역시 존재함을 관찰했다.

5. 결론

본 논문은 single-machine-based 그래프 엔진의 state-of-the-art 모델인 RealGraph 를 이용해 스토리지 서버를 통한 확장가능성을 보였다. 우리는 그래프 엔진이 local NVMe 를 사용할 때와 달리 스토리지 서버를 사용할 때 구조적 변경점이 필요하다는 점을 관찰했으며 IO 제출 thread 를 늘릴 경우 성능향상 가능함을 실험을 통해 확인했다. 이를 통해 향후 연구에서 스토리지 서버를 이용한 single-machine-based 그래프 엔진의 확장을 기대한다.

사사

이 논문은 2023 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2020-0-01373, 인공지능대학원지원(한양대학교)). 또한 이 논문은 2018 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.2018R1A5A7059549). 또한 이 논문은 2023 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.RS-2022-00155586, 실세계의 다양한 다운스트림 태스크를 위한 고성능 빅 하이퍼그래프 마이닝 플랫폼 개발(SW 스타랩))

참고문헌

[1] Lawrence Page, et al., “The pagerank citation ranking: Bring order to the web.", technical report, Stanford University, 1998.

[2] Hilmi Yildirim, and Mukkai S. Krishnamoorthy, "A random walk method for alleviating the sparsity problem in collaborative filtering.", Proceedings of the 2008 ACM conference on Recommender systems, 2008.

[3] Robert Sedgewick and Kevin Wayne, “Algorithms.” Addison-wesley professional, 2011.

[4] Masoud Rehyani Hamedani et al., “AdaSim: A Recursive Similarity Measure in Graphs.”, In ACM CIKM, Australia, 2021, 1528–1537.

[5] Yong-Yeon Jo, et al., "Realgraph: A graph engine leveraging the power-law distribution of real-world graphs.", The World Wide Web Conference, 2019.

[6] Da Zheng, Disa Mhembere, Randal Burns, Joshua Vogelstein, Carey E Priebe, and Alexander S Szalay, “FlashGraph: Processing billion-node graphs on an array of commodity SSDs.”, In Proceedings of the USENIX conference on file and storage technologies (FAST), 2015, 45–58.

[7] Xiaowei Zhu, Wentao Han, and Wenguang Chen, “GridGraph: Large-scale graph processing on a single machine using 2-level hierarchical partitioning.”, In Proceedings of the USENIX annual technical conference (ATC), 2015, 375–386.

[8] Joseph E Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin, “PowerGraph: Distributed graph-parallel computation on natural graphs.”, In Proceedings of the USENIX symposium on operating systems design and implementation (OSDI), 2012, 17–30.

[9] Ching Avery, “Giraph: Large-scale graph processing infrastructure on hadoop.”, In Hadoop Summit, 2011, 5–9.

[10] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos, "Graph evolution: Densification and shrinking diameters.", ACM transactions on Knowledge Discovery from Data (TKDD), 1, 1, 2-es, 2007.

[11] Ziye Yang et al., “SPDK: A development kit to build high performance storage applications.”, In IEEE CloudCom, 2017, 154–161.