고성능 스토리지를 갖는 GPU 기반 그래프 분석 엔진을 위한 I/O 최적화 전략

박정민 [1], 장명환 [2], 김상욱 [3*]
[1] 한양대학교 컴퓨터소프트웨어학과 석사과정
[2] 한양대학교 컴퓨터소프트웨어학과 박사과정
[3] 한양대학교 컴퓨터소프트웨어학과 교수*
{jmpark96, sugichiin , wook}@hanyang.ac.kr

I/O Optimization Strategies for a GPU-based Graph Engine with High-Performance Storage

Jeong-Min Park, Myung-Hwan Jang, Sang-Wook Kim*
Department of Computer Science, Hanyang University

요 약

본 논문은 고성능 스토리지를 사용하는 환경에서 대규모 그래프를 분석을 위한 GPU 기반 그래프 분석 엔진의 I/O 최적화 전략을 제안한다. 사전 실험을 통해 최신 GPU 기반 그래프 엔진인 RealGraph^{GPU} 가 고성능 스토리지의 대역폭을 충분히 활용하지 못하고 있음을 발견하였다. 이를 개선하기 위해 (1) User-space I/O, (2) Asynchronous I/O 두 가지 최적화 전략을 적용하였으며, 실험을 통해 두 전략이 RealGraph^{GPU} 의 그래프 분석 성능 향상시키는 데 효과적임을 확인하였다.

1. 서론

그래프는 소셜 네트워크 분석, 웹 그래프 등 실제 객체 간의 관계를 모델링 하는데 사용하는 데이터 구조이다. 그래프를 분석함으로써, Community detection, Link prediction, Recommendation 등 downstream task 를 처리하기 위한 유용한 지식을 얻을 수 있다. [1,2] 최근, 실제 그래프의 크기는 수백 GB 이상으로 빠르고 지속적으로 증가하고 있다. 이러한 대규모 그래프를 효율적으로 분석하기 위해 그래프 분석 엔진에 대한 많은 연구 노력이 이루어져 왔다. [3,4,5,6,7]

SSD 와 같은 외장 스토리지가 TB 규모의 큰 용량을 제공하게 되면서, 대규모 그래프를 싱글 머신의 스토리지에 저장하고 필요한 그래프를 메인 메모리에 지속적으로 적재하여 처리하는 싱글 머신 기반 그래프 엔진에 대한 연구가 촉진되었다. 싱글 머신 기반 방식은 컴퓨팅 리소스가 제한되어 있음에도 여러 컴퓨터를 동시에 사용하는 분산 시스템 기반 그래프 엔진과 동등하거나 더 나은 성능을 제공하는 경우도 있음을 실험을 통해 입증하고 있다. [3,4,5] 또한, 싱글 머신에서 그래프 분석 속도를 가속하기 위해 GPU 를 활용한 GPU 기반 그래프 엔진이 등장하였으며, [6,7] GPU 기반 방식은 메인 메모리에 적재된 그래프를 GPU 메모리에 적재하여 GPU 가 이를 처리하는 구조를 가지고 있다. 이들 중 RealGraph^{GPU}[7]는 실제 그래프의 특성을 고려하여 그래프 분석 속도를 향상시킨 연구로 기존 싱글 머신 기반 그래프 엔진들보다 월등한 성능을 보였다.

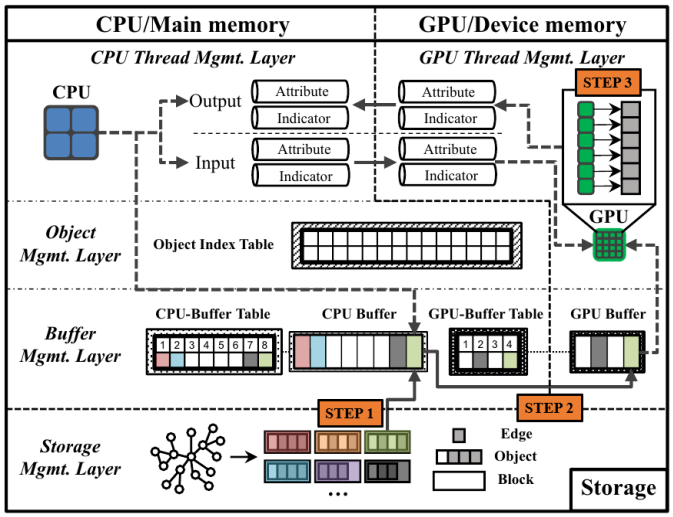
최신 기술의 발전으로 기존 SATA-SSD 보다 10 배 이상의 스토리지 I/O 대역폭을 제공하는 NVMe-SSD 가 개발되면서, GPU 기반 그래프 엔진을 비롯한 싱글 머신 기반 그래프 엔진들은 이러한 하드웨어를 활용하여 스토리지 병목을 완화하는 방식으로 발전해 오고 있다. [6] 이에 따라 본 논문에서, 우리는 최신 그래프 엔진 RealGraph^{GPU} 가 NVMe-SSD 가 제공하는 높은 대역폭을 얼마나 잘 활용할 수 있는지 분석하고자 한다.

2. RealGraph^{GPU}

RealGraph^{GPU} 는 실제 세계의 그래프의 특성인 power-law degree distribution 을 고려하여 설계되었다. RealGraph^{GPU} 는 edge-based workload allocation 을 통한 워크로드 밸런싱과 buffer pre-checking(GPU)을 통한 효율적인 I/O 로 GPU 가 제공하는 높은 병렬처리 능력을 최대한 활용하고자 하였다. RealGraph^{GPU} 의 동작은 iteration 단위로 결과를 생성하며 그래프 알고리즘의 최종 결과가 나올 때까지 iteration 을 수행한다.

그림 1 은 RealGraph^{GPU} 의 5-layer 아키텍처 및 처리 흐름을 보인다. 아키텍처에 대한 설명은 다음과 같다.

* 교신저자



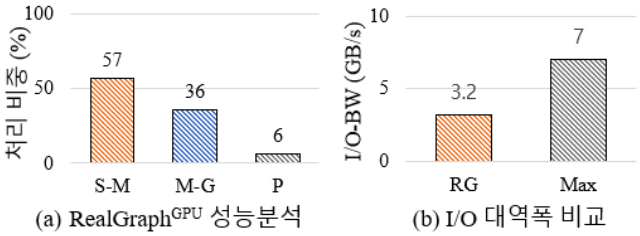
(그림 1) RealGraph^{GPU} 의 아키텍처 및 처리 흐름

- Storage Mgmt. Layer: 스토리지에 저장된 그래프를 고정 크기의 블록으로 관리한다
- Buffer Mgmt. Layer: 메인 메모리(MM) 및 GPU 메모리(GM)에 적재되는 블록을 버퍼 테이블을 사용하여 관리한다.
- Object Mgmt. Layer: 각 블록에 들어있는 그래프에 대한 정보를 관리한다.
- CPU Thread Mgmt. Layer: MM-GM 간 데이터 전송을 담당한다. attribute 는 현재까지의 그래프 알고리즘 수행 결과를 저장하고 indicator 는 다음 동작에서 처리할 그래프에 대한 정보를 저장한다.
- GPU Thread Mgmt. Layer: 그래프 알고리즘 수행, 즉 attribute/indicator 업데이트를 담당한다. 작업이 완료되면 attribute/indicator 를 CPU thread Mgmt. Layer 로 다시 전송한다.

RealGraph^{GPU} 의 Iteration 처리 흐름은 다음과 같다. 먼저 indicator 와 object index 를 기반으로 현재 iteration 에 필요한 그래프를 (1) 스토리지에서 MM 으로 적재(Storage-to-MM I/O) 하고 (2) MM 에 적재된 그래프를 MM 에서 GM 으로 적재(MM-to-GM I/O)한다. 이후에 (3) GM 에 적재된 그래프를 GPU 가 처리(Processing)하여 attribute / indicator 를 업데이트한다. RealGraph^{GPU} 는 pipelining 을 통해 낮은 I/O 속도를 보완하지만 (1) 단계 혹은 (2) 단계의 I/O 속도가 느리다면 (3) 단계에서 처리할 데이터가 없기에 GPU 의 높은 처리 성능을 충분히 활용하기 어렵다.

3. 사전 실험

RealGraph^{GPU} 와 같이 스토리지에 데이터를 저장하고 지속적으로 적재하여 사용하는 시스템의 주요 병목은 Storage-to-MM I/O 이다. [3,4] 그러나 최근 기술의 발전으로 기존 SATA-SSD 보다 10 배 이상 빠른 고대역폭의 스토리지가 개발되었다. 이에 따라 우리는 고성능 스토리지 환경에서 RealGraph^{GPU} 의 스토리지 병목이 해소되었는지 알아보기 위해 사전 실험을 진행하였다. 실험은 RealGraph^{GPU} 가 Pagerank 알고리즘[3]을 수행할 때 세 가지 동작, (1) 1MB 크기의 그래프를 스토리지에서 메인 메모리로 적재(S-M), (2) 메인 메모리에서 GPU 메모리로 적재(M-G), (3) GPU 가 처리(P)에 걸리는 시간을 측정하는 방식으로 진행하였다.



(a) RealGraph^{GPU} 성능분석 (b) I/O 대역폭 비교

(그림 2) 사전 실험 결과

그림 2-(a)는 RealGraph^{GPU} 의 동작 별 처리 비중을 나타낸다. 가장 처리 비중이 높았던 동작은 Storage - to-MM I/O (S-M) 3.2GB/s 였다. 이를 통해 고성능의 스토리지를 사용했음에도 여전히 Storage-to-MM I/O 는 전체 성능의 bottleneck 임을 알 수 있다. 그림 2-(b)는 RealGraph^{GPU} 가 I/O 대역폭을 얼마나 사용하고 있는지 나타낸다. Max 는 NVMe-SSD 가 최대로 제공 가능한 I/O 대역폭이며 RG 는 RealGraph^{GPU} 의 평균 I/O 대역폭을 나타낸다. RealGraph^{GPU} 는 스토리지가 제공하는 I/O 대역폭의 절반도 활용하지 못하는 결과를 볼 수 있다.

사전 실험의 결과로 지금의 RealGraph^{GPU} 가 절반 미만으로 사용하는 고성능 스토리지의 I/O 대역폭의 더 활용할 수 있다면 RealGraph^{GPU} 의 그래프 분석 성능이 크게 향상될 것임을 알 수 있다. 본 논문에서는 RealGraph^{GPU} 에 두 가지 I/O 최적화 전략을 적용하여 고성능 스토리지가 제공하는 I/O 대역폭을 더 많이 사용할 수 있도록 한다.

4. I/O 최적화 전략

본 섹션에서는 RealGraph^{GPU} 가 고성능 스토리지를 사용하는 시스템에서 Storage-to-MM I/O 대역폭을 더 많이 활용할 수 있는 두 가지 전략에 대해 설명한다.

User-space I/O(U-I/O): RealGraph^{GPU} 는 스토리지에서 메모리로 그래프를 적재할 때 Kernel-space I/O (K-I/O) 를 사용한다. K-I/O 는 그래프를 적재할 때마다 user mode 에서 kernel mode 로의 context switching 이 발생하게 되는데, 이 context switching 은 상당한 오버헤드가 된다. [8] 반면, U-I/O 의 경우 context switching 없이 user-space 에서 I/O 를 요청할 수 있다. 따라서 I/O 를 이전보다 빠르게 할 수 있게 되고 이에 따라 I/O 대역폭이 높아지게 된다.

Asynchronous I/O(A-I/O): RealGraph^{GPU} 는 스토리지에서 메모리로 그래프를 적재할 때 Synchronous I/O (S-I/O)를 사용한다. S-I/O 는 thread 가 필요한 그래프에 대해 스토리지에게 적재 요청을 하고, 요청한 그래프가 전부 메모리에 적재될 때까지 다른 일을 하지 않고 기다리는 방식이다. 따라서 thread 는 그래프의 적재가 완료될 때까지 적재 요청을 하지 못하고 대기하게 된다. [3][5] 반면, A-I/O 는 thread 가 스토리지에게 I/O 를 요청한 후 적재 완료까지 대기하지 않고 다른

그래프의 적재를 요청할 수 있다. 따라서, A-I/O 방식으로 동작하는 경우 S-I/O 방식에 비해 thread 가 스토리지에게 적재 요청을 자주 할 수 있게 되고, I/O 대역폭이 높아지게 된다.

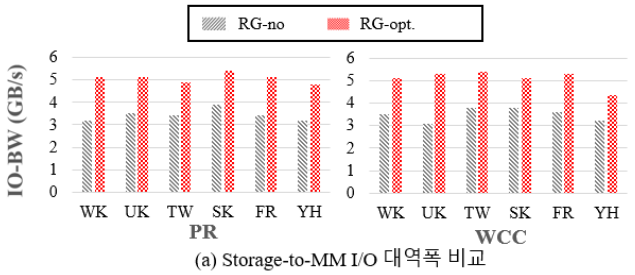
U-I/O 를 사용하기 위해서는 thread 가 kernel 의 도움 없이 스토리지에게 I/O 요청을 해야 한다. 이를 위해서 NVMe-SSD 를 직접 제어하는 NVMe driver 를 kernel space 가 아닌 user space 에서 동작하도록 변경해야 한다. A-I/O 를 사용하기 위해서는 thread 가 스토리지에게 요청한 I/O 가 완료되면 thread 가 완료 여부를 알 수 있도록 polling 방식으로 변경해야 한다.

본 논문에서는 U-I/O 와 A-I/O 를 구현하기 위해 고성능 스토리지 애플리케이션을 위한 library 인 Intel 의 Storage Development Performance Kit (SPDK)[8]를 사용했다.

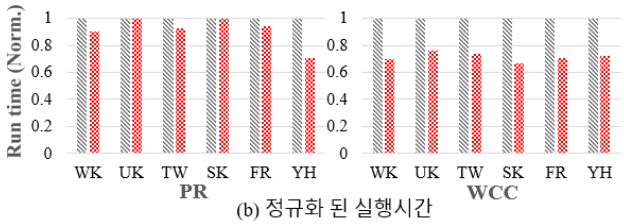
5. 실험

실험 섹션에서는 광범위한 실험을 통해 본 논문에서 제안한 두 가지 I/O 최적화 전략의 효과를 평가한다. 기존 RealGraph^{GPU} (RG-no) 와 최적화 전략을 적용한 RealGraph^{GPU} (RG-opt.)의 대역폭과 수행시간을 비교하고 개선 폭을 관찰한다.

실험은 Intel Xeon 4309Y CPU 2 개, 128GB DRAM, NVIDA A40 GPU, 2TB NVMe-SSD(7GB/s BW) 가 장착된 서버에서 수행하였다. GM 과 MM 은 최대 16GB 로 설정하였다. 6 개의 실제 세계 데이터셋-Wiki, UK, Twitter, SK, Friend, Yahoo[3] 과 유명한 그래프 알고리즘 Pagerank, Weakly Connected Component(WCC)을 사용하여 최적화 전략의 효과를 평가하였다.



(a) Storage-to-MM I/O 대역폭 비교



(b) 정규화 된 실행시간

(그림 3) 실험 결과

그림 3-(a)는 RG-no 에 A-I/O 와 U-I/O 를 적용했을 때 스토리지 대역폭을 얼마나 사용할 수 있는지를 나타낸다. RG-no 의 경우 평균 3493MB/s 의 대역폭을 사용하였지만, 본 논문의 I/O 최적화를 적용한 RG-opt. 의 경우 평균 5208MB/s 의 대역폭을 사용하여, 최대 68%, 평균 42%의 대역폭 향상을 보였다. 그림 3-(b)는 본 논문의 두 가지 최적화 기법이 수행시간에 미치는 영향을 보여준다. 각 알고리즘/데이터셋 별 수행시간의 차이가 큰 관계로 각 결과의 RG-no 수행시간을 1 로 두고 정규화하여 RG-opt.를 나타냈다. 실험 결과 RG-opt.는 RG-no 보다 수행시간 측면에서 최대 57%, 평균 23% 감소하여 성능이 향상되었다. 두 실험을 통해 우리의 최적화 전략은 I/O 대역폭 및 그래프 분석 성능 향상에 효율적임을 관찰했다.

6. 결론

본 논문은 NVMe-SSD 가 스토리지 병목을 해소하기 위해 널리 사용됨에 따라 고성능 스토리지가 제공되는 환경에서 기존 최신 그래프 엔진 RealGraph^{GPU} 의 bottleneck 을 파악하고 이를 해결하기 위해 두 가지 최적화 기법(A-I/O, S-I/O)을 적용하였다. 실험을 통해 본 논문에서 두 가지 최적화 전략 대역폭과 수행시간 측면에서 성능이 향상됨을 보였다.

사사

본 연구는 삼성전자와 한양대의 산학협력의 일환으로 수행되었음. 또한 본 연구는 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)). 또한 본 연구는 2018 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No.2018R1A5A7059549).

참고문헌

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

[2] Yoonsuk Kang et al. Cr-graph: Community reinforcement for accurate community detection. In ACM CIKM. Online. 2020. 2077–2080.

[3] Yong-Yeon Jo et al. RealGraph: A graph engine leveraging the power-law distribution of real-world graphs. In ACM WWW. USA. 2019. 807–817.

[4] Aapo Kyrola et al. GraphChi: Large-scale graph computation on just a pc. In USENIX OSDI. USA. 2012. 31–46.

[5] Wook-Shin Han et al. TurboGraph: A fast parallel graph engine handling billion-scale graphs in a single PC. In ACM KDD. USA. 2013. 77–85.

[6] Min-Soo Kim et al. GTS: A Fast and Scalable Graph Processing Method based on Streaming Topology to GPUs. In ACM SIGMOD. USA. 2016. 447-461.

[7] Myung-Hwan Jang et al. RealGraph^{GPU}: A High-Performance GPU-Based Graph Engine toward Large-Scale Real-World Network Analysis. In ACM CIKM. USA. 2022. 4074-4078.

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