랜덤워크 기법을 위한 GPU 기반 희소행렬 벡터 곱셈 방안에 대한 성능 평가

유재서, 배홍균, 강석원, 유용승, 박영준, 김상욱†
한양대학교 컴퓨터소프트웨어학과
{yjs08090, hongkyun, kswon0202, dydtmd1991, yongjunpark, wook}@hanyang.ac.kr

GPU-based Sparse Matrix-Vector Multiplication Schemes for Random Walk with Restart: A Performance Study

Jae-Seo Yu, Hong-Kyun Bae, Seokwon Kang, Yongseung Yu, Yongjun Park, Sang-Wook Kim
Dept. of Computer Science, Hanyang University

요 약

랜덤워크 기반 노드 랭킹 방식 중 하나인 RWR(Random Walk with Restart) 기법은 희소행렬 벡터 곱셈 연산과 벡터 간의 합 연산을 반복적으로 수행하며, RWR 의 수행 시간은 희소행렬 벡터 곱셈 연산 방법에 큰 영향을 받는다. 본 논문에서는 CSR5(Compressed Sparse Row 5) 기반 희소행렬 벡터 곱셈 방식과 CSR-vector 기반 희소행렬 곱셈 방식을 채택한 GPU 기반 RWR 기법 간의 비교 실험을 수행한다. 실험을 통해 데이터 셋의 특징에 따른 RWR 의 성능 차이를 분석하고, 적합한 희소행렬 벡터 곱셈 방안 선택에 관한 가이드라인을 제안한다.

1. 서론

최근, 클릭 또는 구매를 비롯한 다양한 유저 행동 정보의 양이 빠르게 늘어남에 따라, 추천 시스템의 추천 결과 도출에 소요되는 시간 또한 크게 증가하고 있다. 이러한 이유로 GPU(Graphics Processing Unit)를 기반으로 하는 연산 기법에 관한 연구들이 활발하게 수행되고 있다. GPU 는 단순한 연산의 반복으로 이루어진 방대한 양의 작업을 병렬적으로 처리하기 때문에, 연산에 소요되는 시간을 크게 줄일 수 있다.

랜덤워크 기반 노드 랭킹 방법인 RWR(Random Walk with Restart)은 널리 사용되는 그래프 기반 추천 기법 중 하나이다[1]. RWR 은 희소행렬 벡터 곱셈(Sparse Matrix-Vector Multiplication) 연산과 두 벡터 간의 합 연산을 반복적으로 수행한다. 하지만 일반적으로 희소행렬 상에서 불규칙한 데이터 접근 패턴은 GPU 를 통한 RWR 성능 향상을 가로막는 요인이 된다. 이러한 문제를 해결하기 위해 희소행렬 벡터 곱셈에 대한 여러 기법들이 연구되어 왔다. 그 중, CSR(Compressed Sparse Row) 기반 행렬 곱셈 방식인 CSR-vector 기법[2]과, CSR5(Compressed Sparse Row 5)[3] 기반 희소행렬 곱셈 기법은 GPU 에서 희소행렬을 효과적으로 처리하기 위해 제안된 대표적인 방법이다.

본 논문에서는 CSR-vector 및 CSR5 기반 희소행렬 곱셈 기법을 사용하여 RWR 을 가속화하고, 이들에 대한 성능 분석을 통하여 RWR 에 적합한 희소행렬 곱셈 방식 선택에 대한 통찰을 제공한다.

2. 희소행렬 곱셈 방안

CSR 포맷은 가장 유명한 희소행렬 표현 방법 중 하나이다[2]. CSR 포맷에 따라 행렬은 열의 인덱스를 저장하는 배열, 0 이 아닌 값들을 저장하는 배열, 각 행의 시작 주소를 나타내는 포인터 배열을 합쳐 총 세 가지의 정보로 표현된다. 워프(Warp)란 스트리밍 멀티프로세서(Streaming Multiprocessors)의 스레드 스케줄링 단위이다. CSR-vector 기법은 각 워프를 CSR 포맷으로 표현된 그래프 데이터의 각 행 마다 할당하여 희소행렬 곱셈 연산을 수행한다.

CSR-vector 기법은 연속적으로 메모리에 저장된 각 열의 인덱스 배열과 0 이 아닌 값을 저장하는 배열을 워프 내의 스레드가 동시에 접근하게 만듦으로써, 높은 메모리 처리량(throughput)을 얻을 수 있다는 장점이 있다[2]. 또한, 희소행렬 곱셈을 위한 별도의 포맷 변환이 필요하지 않다. 단, 행렬의 각 행당 0 이 아닌 값의 수 차이가 큰 경우, 워크로드 밸런싱(Workload Balancing)의 수준이 낮아져 연산 수행 성능이 제한될 수 있다.

CSR5 포맷은 희소행렬을 일정 크기의 타일로 분할한다. CSR 의 세 가지 정보와 함께 각 타일의 시작

† 교신저자

주소 및 희소행렬 곱셈 연산에 필요한 두 가지 정보를 추가적으로 담고 있다. 추가적인 정보 외에 기존 CSR 포맷의 열의 인덱스와 0 이 아닌 값을 저장하는 배열을 타일 기준으로 전치하여 저장한다.

곱셈 연산 시에는 각 워프를 타일 마다 할당하는데, 각 워프는 동일한 크기의 타일을 처리하기 때문에 워크로드 밸런싱 효과를 얻을 수 있다. 또한, 가장 흔히 사용되는 CSR 포맷을 기반으로 한다는 점에서, 희소행렬의 포맷을 크게 변형시키는 기타 희소행렬 벡터 곱셈 방식에 비하여 데이터 변형으로 인한 추가 비용이 적다는 장점이 있다. 하지만 타일링 및 곱셈을 위한 추가 데이터가 필요하며, 전치로 인한 오버헤드가 존재한다[4].

3. 성능 평가

3.1 실험 환경

본 논문에서는 추천시스템에서 자주 사용되는 세 가지의 데이터 셋을 사용하여 실험을 수행하였다. 표 1 은 사용한 데이터 집합의 통계를 보여준다.

표 1. 데이터 집합의 통계

데이터	평점	유저	아이템	유저 당 평점 평균	희소도
ML10M	10,000,054	71,567	10,681	143	98.7%
Amazon	22,507,155	8,026,324	2,330,066	2.8	99.9%
Netflix	100,480,507	480,189	17,770	209	98.8%

실험은 Ubuntu 18.04 운영체제에서 수행하였으며, CPU 는 Intel core i9-9900k, GPU 는 Nvidia Geforce RTX 2070 을 사용하였다. 연산장치 상세 정보는 표 2 에 나와있으며, 개방형 범용 컴퓨팅 프레임워크인 OpenCL 1.2 를 사용하였다.

표 2. 연산장치 정보

튜링(Turing)플랫폼	
중앙처리장치	인텔 옥타 코어 CPU (16 스레드), 16GB 메모리
그래픽처리장치	RTX 2070: 36 SMs, 8GB 디바이스 메모리, 448 GB/s 메모리 대역폭, SM 당: 16 쿠다(CUDA) 코어 , 2K 스레드

3.2 실험 결과

실행 시간 측정 실험 비교 대상은 GPU CSR5 RWR, GPU CSR-vector RWR, CPU CSR RWR 총 세 가지 버전이다. CPU CSR RWR 을 베이스라인으로 설정하였으며, 단일 스레드로 실행된다. 10 명의 유저에 대하여 특정 유저를 위한 RWR 스코어 벡터를 구할 때 소요되는 평균 시간을 밀리초(millisecond) 단위로 측정하였고, 결과를 RWR 총 소요시간, 희소행렬 벡터 곱셈 시간 (CSR5 의 경우 포맷 시간 포함), 포맷 변환 시간으로 구분하여 표 3 에 기록하였다.

실험 결과를 보면, GPU RWR(CSR5, CSR-vector) 버전의 속도는 CPU RWR 에 비하여 약 4~47 배 더 빠르다.

GPU CSR5 RWR 의 경우 Netflix, Amazon 데이터 셋에서 가장 좋은 성능을 보이고, GPU CSR-vector RWR 의 경우 작은 데이터 셋인 ML 10M(Movielens 10M)에서 우수한 성능을 보인다. 작은 데이터 셋에서 GPU CSR5 RWR 의 성능이 좋지 못한 이유는 CSR 에서 CSR5 로의 포맷 변환 시간이 전체 실행 시간 중 차지하는 비중이 크기 때문이다.

유저 당 평균 평점 수가 적은 Amazon 데이터 셋에서는 CSR5 를 사용한 RWR 의 성능 개선 폭이 CSR-vector RWR 의 개선 폭에 비해 크다는 것을 알 수 있다. 타일 구조에 의해 워크로드 밸런싱이 상대적으로 잘 이뤄지는 CSR5 방식에 비해서, CSR-vector 방식에서는 0 이 아닌 값을 워프 내의 스레드 수(=32)보다 적게 갖는 행이 많을수록 성능이 쉽게 저하될 수 있기 때문이다.

표 3. 실행 시간 비교 (밀리초)

데이터	버전	총 시간	곱셈	포맷 변환
ML10M	GPU CSR5	19.57(42.7x)	14.85(27.6x)	1.06
	GPU CSR-vector	19.46(42.9x)	14.74(27.8x)	-
	CPU CSR	834.6(1x)	409.84(1x)	-
Amazon	GPU CSR5	396.58(10.3x)	133.25(24.7x)	7.3
	GPU CSR-vector	912.89(4.5x)	649.55(5.1x)	-
	CPU CSR	4,064.43(1x)	3,285.43(1x)	-
Netflix	GPU CSR5	240.88(47x)	197.47(27.1x)	8.81
	GPU CSR-vector	244.61(46x)	210(26.6x)	-
	CPU CSR	11,351.54(1x)	5,596.3(1x)	-

4. 결론

본 논문에서는 대표적인 그래프 기반 추천 방식인 RWR 기법에 대한 GPU 기반 희소행렬 벡터 곱셈 방안들 간 성능을 비교하였다. 일반적으로 CSR-vector 방식의 단점을 보완한 CSR5 기반 희소행렬 곱셈이 더 우수한 것으로 알려져 있다. 하지만 (1) 크기가 상대적으로 작은 데이터 셋에서는 별도의 포맷 변환 과정이 요구되지 않은 CSR-vector 방식이 더 좋은 성능을 보였다. 반면 (2)데이터의 희소도가 상대적으로 더 큰 Amazon 데이터 셋에서는 CSR5 방식이 더 우수한 성능을 보인 것을 알 수 있었다. 따라서 RWR 을 가속화할 때 그래프 데이터의 특성을 분석하여 CSR5, CSR-vector 방식 중 적합한 방식을 선택해야 한다.

사사

이 논문은 삼성전자 미래기술육성센터의 지원을 받아 수행된 연구 (과제번호 SRFC-IT1901-03)임.

참고문헌

[1] Konstas, V. Stathopoulos, and J. M. Jose, “On social networks and collaborative recommendation,” in Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval. ACM, 2009, pp. 195–202.

[2] N. Bell and M. Garland. Implementing Sparse Matrix-Vector Multiplication on Throughput-Oriented Processors. In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09, pages 18:1-18:11, 2009.

[3] W. Liu and B. Vinter. Csr5: An efficient storage format for cross-platform sparse matrix-vector multiplication. In ICS’15. ACM, 2015.

[4] Salvatore Filippone, Valeria Cardellini, Davide Barbieri, and Alessandro Fanfarillo. 2017. Sparse matrix-vector multiplication on GPGPUs. ACM Trans. Math. Software 43, 4, Article 30 (Jan. 2017), 49 pages.