대규모 그래프 파티셔닝을 위한 Warp 협력 기반 라벨 전파 최적화

김민창, 윤태영, 추민솔, 오상윤*
아주대학교

minchang111@ajou.ac.kr, tree030720@ajou.ac.kr, cmss1217@ajou.ac.kr, syoh@ajou.ac.kr*

Warp-Cooperative Label Propagation Optimization for Large-Scale Graph Partitioning

Minchang Kim, Taeyoung Yun, Minsol Choo, Sangyoon Oh*
Ajou University

요 약

본 논문은 GPU 기반 그래프 파티셔닝에서 라벨 전파 수행 시 발생하는 비효율적인 메모리 접근과 스레드 간 부하 불균형 문제를 개선하는 기법을 제안한다. 제안 기법은 소속 변경 가능성이 있는 경계 정점만 추출하여 서브 그래프를 구성함으로써 메모리 접근 지역성을 향상하고, 하나의 Warp 가 하나의 정점을 협력 처리하도록 커널을 설계하여 스레드 간 작업량을 균등 분배한다. 실험을 통해 제안 기법이 기존 방식 대비 실행 시간을 60.4% 단축하고 Edge cut 을 17.6% 감소시켜, 제한된 GPU 자원 내에서 연산 효율을 유의미하게 개선됨을 확인하였다.

Ⅰ. 서 론

소셜 네트워크, 웹 그래프, 생물학적 네트워크 등 현대의 그래프 데이터는 수십억 개의 정점과 에지를 포함하는 경우가 많다. 이러한 대규모 그래프는 단일 노드의 메모리 용량과 처리 능력을 초과하므로 여러 노드에 분산하여 처리해야 한다. 이때 그래프를 어떻게 나누는가에 따라 노드 간 통신량이 달라지는데, 파티션간 연결 에지가 많을수록 통신비용이 증가하여 전체 처리 성능이 저하된다. 특히 그래프 파티셔닝의 분할 및 정제 단계에서는 각 노드 내부에서 라벨 전파를 반복적으로 수행하며, 이 단계의 GPU 효율이 전체 파티셔닝 시간에 큰 영향을 준다.

그래프 파티셔닝은 파티션간 연결 에지 수를 최소화하면서 각 파티션의 크기를 균형 있게 유지하는 것을 목표로 한다. 여기서 균형은 정점 수 기준의 Vertex Balance 와 에지 수 기준의 Edge Balance 로 측정한다. 이 문제를 해결하기 위해, Kernighan-Lin [1]과 같은 알고리즘이 제안되었으나, 이들은 정점 교환 과정에서 이전 교환 결과를 반영해야 하므로 본질적으로 순차 처리가 필요하다.

반면, 라벨 전파는 각 정점이 자신의 이웃들이 속한 파티션 정보만 확인하여 다수가 속한 파티션으로 소속을 변경하는 방식으로, 정점 간 연산이 독립적이어서 대규모 병렬 처리에 적합하다. 그러나 정점 당 스레드 하나를 할당하는 일반적인 방식은 이웃이 많은 정점의 처리가 지연되어 동일 워프 내 다른 스레드들이 유휴 대기 상태가 되고, 결과적으로 심각한 부하 불균형을 발생시킨다. 또한 정점 ID 순서대로 메모리에 배치된 그래프 구조에서는 실제 연산에 필요한 정점들이 메모리상에 흩어져 있어 캐시 효율이 저하된다.

본 논문은 이러한 문제들을 개선하기 위해 두 가지 기법을 제안한다. 첫째, 소속 변경 가능성이 있는 경계 정점만 식별하여 서브 그래프를 구성하고 CSR (Compressed Sparse Row) 형식으로 재배열함으로써 메모리 참조의 지역성을 최적화한다. 이는 전체 그래프를 무작위로 참조할 때 발생하는 캐시 미스를 줄여준다. 둘째, 하나의 Warp 가 하나의 정점을 협력 처리하도록 커널을 설계하여 이웃 수와 관계없이 스레드 간 작업량을 균등하게 분배함으로써 Warp 내 연산 효율을 극대화한다.

Ⅱ. 관련 연구

METIS [2]는 그래프 축소, 초기 파티셔닝, 파티션 복원의 다단계 구조를 사용하는 대표적인 파티셔닝 도구이다. 축소 단계에서 유사한 정점들을 병합하여 그래프 크기를 줄이고, 작아진 그래프에서 파티셔닝을 수행한 뒤, 복원 단계에서 원래 크기로 되돌리면서 경계를 정제한다. 다만, METIS 는 높은 품질의 파티션을 생성하지만, CPU 기반 순차 처리에 의존하여 대규모 그래프에서는 처리 시간이 오래 걸린다.

Jet [3]은 GPU 에서 다단계 그래프 파티셔닝을 수행하며, 복원 단계에서 라벨 전파 기반의 파티션 정제를 병렬로 처리한다. 그러나 Jet 은 전체 그래프를 GPU 메모리에 적재하는 방식이므로 메모리 지역성 측면에서 개선의 여지가 있다. 또한 정점 당 스레드 하나를 할당하는 구조로 인해 고 차수 정점에서 부하 불균형이 발생할 수 있다.

Ⅲ. 제안 기법

3.1 경계 정점 중심 서브 그래프 구성

라벨 전파에서 다른 파티션에 속한 이웃을 가진 경계 정점만이 소속 변경 가능성이 있어 실제 연산 대상에 해당한다. 본 기법은 경계 정점과 그 1-hop 이웃만을 추출하여 서브 그래프를 구성하고, CSR 형식으로 재배열하여 메모리 접근 지역성을 향상한다.

서브 그래프 구성에는 정점 식별 및 재배열 오버헤드가 수반되나, 멱 법칙 분포 특성상 경계 정점 비중이 작아 구성 비용이 최소화된다. 또한, 서브 그래프는 원본 인접 리스트의 인덱스 매핑만 유지하고, 경계 정점에 대해서만 투표 집계 버퍼를 할당하므로 추가 메모리 오버헤드가 제한적이다. 경계 정점 집합은 매 반복 갱신되나, 이러한 낮은 오버헤드로 인해 전체 실행시간을 효과적으로 단축할 수 있다.

3.2 Warp 협력 처리

GPU 의 Warp 는 32 개 스레드가 동일한 명령어를 동시에 실행하는 단위이다. Warp 내부 스레드들은 분기가 발생하면, 가장 오래 걸리는 스레드가 완료될 때까지 나머지 스레드들은 대기해야 한다. 따라서 정점마다 스레드 하나를 할당하는 방식은 이웃이 많은 고 차수 정점 처리에 따라 Warp Divergence 와 부하 불균형이 발생한다.

본 기법은 경계 정점 하나를 Warp 하나가 전담하도록 커널을 설계하여 이를 해결한다. 32 개 스레드가 각 정점의 이웃들을 분담하고 병렬로 순회하며, 각 이웃이 속한 파티션 정보를 Warp 별 공유 메모리에 집계한다. 이후 각 라벨에 속한 이웃 수를 기반으로 점수를 계산하고, 가장 높은 점수의 라벨을 선택하기 위해 Warp Shuffle Reduction 을 수행한다.

Warp Shuffle Reduction 은 레지스터 간 직접 데이터 교환을 통해 Warp 내 32 개 스레드의 결과를 5 단계 ($= \log_2 32$)의 병렬 비교만으로 집계하는 연산이다. 이 과정에서 추가적인 메모리 접근이나 Atomic 연산이 필요하지 않아 메모리 경합을 방지하고 연산 효율을 높인다. 또한 이웃이 32 개를 초과하더라도 스레드별 작업량이 균등하게 분배되어 부하 균형이 유지된다.

Ⅳ. 실험 및 결과

실험은 2 개의 노드로 구성된 클러스터 환경에서 수행되었으며, 각 노드에는 NVDIA RTX 3080 GPU 2 개가 장착되어 총 4 개의 GPU 를 사용하였다. 4 개의 프로세스가 각각 하나의 GPU 를 담당하도록 매핑하였으며, 초기 파티션은 정점 개수 기준 균등 분배 방식으로, 8 개로 분할하였다. 본 실험에서는 프로세스 간 통신(MPI)은 기존 방식을 그대로 사용하고, 각 프로세스 내부의 GPU 라벨 전파 커널에 제안 기법을 적용하여 성능을 측정하였다. 라벨 전파의 최대 반복 횟수는 데이터셋의 특성에 따라 100~500 회로 설정하였으며, 이전 반복 대비 라벨 변경 정점 비율이 0.1% 미만일 때 수렴한 것으로 판단하여 종료하였다. 모든 결과는 10 회 독립 실행의 평균값이다.

표 1. Copter2 데이터셋 GPU 커널 성능 비교

항목	기존 기법	제안 기법
실행 시간	183.7 ms	72.8 ms
메모리 사용량	199,228 KB	189,012 KB
SM Occupancy	13.4%	16.6%
Warp Efficiency	41.8%	58.9%
Throughput	0.88 GB/s	13.04 GB/s

표 1 은 copter2 데이터셋에서 측정한 GPU 커널 성능 비교 결과이다. 제안 기법은 실행 시간을 183.7ms 에서 72.8ms 로 60.4% 단축하였다. 특히 DRAM Throughput 이 약 14.8 배 상승하였는데, 이는 전체 그래프가 아닌, 경계 정점 중심의 서브 그래프를 CSR 형식으로 재구성하여 메모리 접근 지역성을 향상시킨 결과로 분석된다. 메모리 사용량은 199,228 KB 에서 189,012 KB 로 약 5% 감소하였는데, 이는 경계 정점에 대해서만 투표 집계 버퍼를 할당하고 Warp 내 레지스터를 활용하여 전역 메모리 할당을 최소화하였기 때문으로 판단된다.

표 2. ljournal-2008 데이터셋 파티셔닝 품질 비교

항목	적용 전	적용 후
Edge cut	20,773,079	17,112,543
Edge Balance	1.75	1.34
Vertex Balance	1.05	1.13

표 2 는 ljournal-2008 데이터셋에서의 파티셔닝 품질 비교 결과이다. 제안 기법 적용 후 Edge cut 은 20,773,079 에서 17,112,543 으로 17.6% 감소하였다. Edge Balance 는 1.75 에서 1.34 으로 23.4% 개선되었다. Node Balance 는 1.05 에서 1.13 로 7.1% 증가하였으나, 대규모 그래프 파티셔닝의 병목지점인 Edge cut 을 줄이는 과정에서 발생하는 트레이드오프의 결과로 해석된다.

Ⅴ. 결론

본 논문에서는 경계 정점 중심 서브 그래프 구성과 Warp 협력 처리를 통해 GPU 라벨 전파의 메모리 지역성과 부하 균형을 개선하였다. 메시 그래프와 소셜 네트워크 그래프 모두에서 실행 시간 단축 및 파티셔닝 품질 향상을 확인하였다. 향후 다중 GPU 환경에서 대규모 그래프 처리로 확장할 계획이다.

ACKNOWLEDGMENT

본 연구는 2026 년 과학기술정보통신부 및 정보통신기획평가원의 SW 중심대학사업(2022-0-01077) 및 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(RS-2023-00283799).

참 고 문 헌

[1] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," Bell System Technical Journal, vol. 49, no. 2, pp. 291-307, 1970.

[2] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," SIAM Journal on Scientific Computing, vol. 20, no. 1, pp. 359-392, 1998.

[3] M. Gilbert, K. Madduri, E. Boman, and S. Rajamanickam, "Jet: Multilevel Graph Partitioning on GPUs," SIAM Journal on Scientific Computing, vol. 46, no. 5, pp. B700-B724, 2024.