분산 환경에서 그래프 질의 수행을 위한 그래프 분할 기법 조사

이원석*, 고성윤*, 서명원*, 이정훈*, 한욱신**
*포항공과대학교 창의 IT 융합공학과
**포항공과대학교 컴퓨터공학과/창의 IT 융합공학과
e-mail : {wslee, syko, mwseo, jhlee, wshan}@postech.ac.kr

A Study on Graph Partitioning for Graph Query Processing in Distributed System

Wonseok Lee*, Seoungyun Ko*, Myeongwon Seo*, Jeong-Hoon Lee*, Wook-Shin Han**
*Dept. of Creative IT Engineering, POSTECH
**Dept. of Computer Science and Engineering/Dept. of Creative IT Engineering, POSTECH

요 약

그래프 분할 기법은 분산 환경에서 그래프 질의 수행에 있어 통신 비용을 줄이고 부하 균형을 맞추고자 그래프의 정점과 간선들을 여러 머신들에 나누어 저장하는 방법이다. 본 논문에서는 그래프 질의 수행에 관한 지식을 정리하고, 간선 절단 기법(edge-cut), 정점 절단 기법(vertex-cut), 하이브리드 절단 기법(hybrid-cut)으로 알려진 대표적인 그래프 분할 기법과 최신 그래프 시스템들의 그래프 분할 기법을 소개하고 비교한다.

1. 서론

본 논문에서는 대표적으로 알려진 그래프 분할 기법과 최신 그래프 처리 시스템들의 분할 기법을 조사하고 비교하였다. 그래프 분할 기법은 분산 환경에서 질의 수행을 위해 그래프의 정점과 간선들을 여러 머신들에 나누어 저장하는 방법을 의미한다. 그래프 분할 기법은 분산 환경에서 머신 간 통신 비용을 줄이고 부하 균형을 맞추어 그래프 질의 수행의 성능을 높이는 데 목적이 있다.[6]

그래프 질의 수행 성능을 높이기 위해 다양한 그래프 분할 기법들이 제안되어 왔다. 대표적으로 알려진 분할 기법으로는 간선 절단 기법(edge-cut), 정점 절단 기법(vertex-cut), 하이브리드 절단 기법(hybrid-cut) 등이 있다. 최신 그래프 처리 시스템인 TopoX[6]와 Gemini[13] 또한 각각 융합과 분열(fussion&fission) 방법을 적용한 하이브리드 절단 기법과 덩어리 기반 절단 기법(chunk-based graph partitioning)을 제시하였다.

본 논문의 구성은 다음과 같다. 2 절에서는 그래프 질의 수행 모델과 그래프 분할에 대한 배경 지식을 정리한다. 3 절에서는 대표적인 그래프 분할 기법인 간선 절단 기법, 정점 절단 기법, 하이브리드 절단 기법과, 최신 분산 그래프 처리 시스템인 TopoX 와 Gemini 의 그래프 분할 기법을 소개한다. 4 절에서는 각 그래프 분할 기법의 성능을 비교하고, 5 절에서 본 논문에 대한 결론을 내린다.

이 논문은 2019 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2017R1A2B3007116)

2. 배경지식

본 절에서는 그래프 질의 수행에 관련된 모델과 그래프 분할에 관련된 배경 지식을 정리한다.

2.1. 분산 환경에서 그래프 질의 수행 모델

그래프에 대한 질의를 수행하기 위한 다양한 모델들이 제안되었다. 그래프 질의 수행 프로그래밍 모델로는 수집-적용-분산(Gather-Apply-Scatter, GAS) 모델[3]이 있다. 수집 단계에서는 수집 함수와 합계 함수를 이용해 각 정점의 들어오는 간선들로부터 소스 정점에서 보낸 정보를 얻고, 이를 합하여 정점마다 한 개의 합해진 결과를 얻는다. 적용 단계에서는 적용 함수를 통해 각 정점마다 합해진 결과로부터 각 정점에 대한 값을 계산하여 저장한다. 분산 단계에서는 각 정점의 나가는 간선(out-edge)들에 새로운 값들을 계산하여 전달한다.

질의 수행 모델로는 정점 중심(vertex-centric), 간선 중심(edge-centric), 블록 중심(block-centric), 이웃 중심(neighborhood) 처리 모델이 있다. 정점 중심 처리 모델은 Pregel[8]이 제안하였으며, 정점 단위로 그래프에 대한 연산을 정의한다. Pregel, PowerGraph[3], Gemini 등이 정점 중심 처리 모델을 사용한다. 간선 중심 처리 모델은 간선 단위로 그래프에 대한 연산을 정의한다. 대표적으로 Chaos[9]가 간선 중심 처리 모델을 사용한다.

한편, 소스(source) 정점의 정보를 타겟(target) 정점에 어떻게 전달할지에 따라 그래프 질의 수행 모델을 푸쉬(push) 모델과 풀(pull) 모델로 나눌 수 있다. 푸쉬 모델에서는 소스 정점에서 나가는 간선의 타겟 정점으로 정보를 전달하고, 풀 모델에서는 타겟 정점에서 들어오는 간선의 소스 정점으로부터 정보를 가져온다. Gemini 와 HybridGraph[12]는 질의 처리 과정 도중에 두 가지 모델 중에서 적절한 모델을 선택하여 그래프 질의 수행의 성능을 높인다. PowerLyra[1]는 타겟 정점의 차수에 따라 간선마다 푸쉬 모델로 처리할지 풀 모델로 처리한다.

마지막으로, 분산 머신들 간의 동기화에 대한 모델로 벌크동기병렬 모델(bulk synchronous model, BSP)과 비동기 병렬 모델(asynchronous model)이 있다. 벌크동기병렬 모델을 사용하는 그래프 처리 시스템은 Gemini 와 Chaos 가 있고, 비동기 병렬 모델을 사용하는 시스템은 GraphChi[5]와 GraphLab[7]이 있다.

2.2. 그래프 분할

그래프 분할 기법의 목적은 분산 환경의 머신 간 통신 비용을 줄이고 부하 균형을 맞추어 그래프 질의 수행의 성능을 높이는 것이다. 최적의 그래프 분할 기법을 찾는 문제는 NP-hard 로 알려져 있으며[6], 이 때문에 다양한 휴리스틱 그래프 분할 기법들이 제안되어 왔다.

분산 환경에서 그래프 질의를 수행하는데 있어 효율적으로 통신 비용을 줄이고 부하 균형을 맞추기 위해서는 그래프의 특성을 고려하여 그래프를 분할해야 한다. 예를 들어, 많은 현실 그래프들은 정점의 차수가 멱분포(power-law distribution)를 따른다. 이런 특성은 균일하게 정점을 나누었을 때 부하의 불균형을 야기한다.[3] 후술할 정점 절단 기법과 하이드리드 절단 기법 및 TopoX 의 그래프 분할 기법은 이러한 특성을 고려하여 그래프를 분할한다. 몇몇 현실 그래프들이 따르는 또다른 특성으로는 토폴로지(topology) 상으로도 지역성을 가지는 정점들이 정점의 식별자(ID)에서 지역성(locality)을 가지지는 것인데, 후술할 Gemini 의 그래프 분할 기법이 이러한 특성을 이용한다.

3. 그래프 분할 기법

본 절에서는 대표적으로 알려진 그래프 분할 기법과 최신 시스템에서 제안한 그래프 분할 기법을 소개한다.

3.1. 간선 절단 기법

간선 절단 기법은 정점들을 분산 머신에 균등하게 분배하고 상응하는 간선들을 정점에 따라 분배하는 기법이다. 간선 절단 기법은 멱분포를 따르는 현실 그래프에서는 부하 균형을 맞추기 어렵다는 단점이 있다.[3] 이 기법을 사용하는 시스템으로는 Pregel, GraphLab, Weaver[2], Wukong[10]이 있다. Pregel 과 GraphLab 은 그래프를 분할하기 어려울 경우 랜덤 해시를 통해 정점을 분배하는데 이 경우 절단되는 간선이 많아져 통신 비용이 커질 수 있다.[3]

3.2. 정점 절단 기법

정점 절단 기법은 간선들을 분산 머신에 균등하게 분배하고 상응하는 시작 정점과 끝 정점을 분배하는 기법이다. 이 기법을 사용하는 시스템으로는 PowerGraph 와 GraphX[4]가 있다. PowerGraph 는 네트워크 통신 비용을 최소화하고자 각 정점이 분배되는 머신 수에 제한을 두는 최적화를 적용한다.[3]

3.3. 하이브리드 절단 기법

하이브리드 절단 기법은 위 두 기법을 결합하여 낮은 차수를 가진 정점에 간선 절단 방식을, 높은 차수를 가진 정점에 정점 절단 방식을 적용하는 기법이다. 이 기법을 사용하는 시스템으로는 PowerLyra 가 있다. PowerLyra 는 낮은 차수의 정점을 분배할 때 이미 분배된 낮은 차수의 정점들이 최대한 많이 이웃하게 하되 부하 균형을 고려하여 분배하는 최적화를 적용한다.

3.4. TopoX 의 분할 기법

TopoX 는 융합과 분열 방법을 통해 그래프를 리팩토링한 후 PowerLyra 의 하이브리드 절단을 적용한다. 융합은 차수가 낮은 정점들을 상위 정점(super-vertex)으로 만드는 과정으로, 임의로 차수가 낮은 정점을 고른 후 해당 정점으로부터 일정 거리 이내의 차수가 낮은 정점들을 모아 상위 정점으로 만든다. 분열은 차수가 높은 정점을 하위 정점(sub-vertex)들로 만드는 과정이다.

3.5. Gemini 의 분할 기법

Gemini 의 덩어리 기반 분할 기법(chunk-based graph partitioning)은 정점의 집합을 연속된 덩어리(chunk)들로 나누어 분할한다. 이 때 부하 균형을 위해 각 덩어리 의 정점의 수와 들어오는 간선 수의 가중 합계의 균형이 맞도록 덩어리들을 나눈다.

Gemini 는 앞서 설명한 그래프 분할 기법들과 달리 한 개의 간선을 타겟 정점으로 들어오는 간선과 소스 정점에서 나가는 간선으로 나누어 각각 분할한다. 이 때 들어오는 간선은 소스 정점의 분할에, 나가는 간선은 타겟 정점의 분할에 분할한다. 이는 Gemini 의 그래프 질의 수행 모델이, 각 단계에서 활성 정점 수에 따라 푸쉬 모델과 풀 모델을 선택하기 때문이다.

4. 성능 평가

본 절에서는 3 절에서 표현한 그래프 분할 기법들을 비교하기 위하여 PowerLyra, TopoX, 그리고 Gemini 논문에서 실험한 결과들을 소개한다. 시스템의 그래프 질의 수행 모델의 영향을 배제하기 위하여, 해당 실험이 없는 Gemini 시스템을 제외하고는 통신 비용과 관련이 있다고 알려진[11] 복제 인자를 이용하여 그래프 분할 기법들을 비교한다. Gemini 의 경우 복제 인자에 관한 실험이 없기 때문에, Gemini 논문의 실행 시간 비교 실험을 소개한다.

PowerLyra 논문에서는 정점 절단 기법과 하이브리

드 절단 기법에 대해 복제 인자의 수를 비교하였다. 멱분포에 따라 생성한 그래프에 대해서는 하이브리드 절단 기법이 정점 절단 기법에 비해 낮은 복제 인자를 보여주었으며, 현실 그래프에 대해서도 최적화를 적용한 PowerLyra 의 하이브리드 절단 기법이 대부분의 경우에서 가장 낮은 복제 인자를 보여주었다. 자세한 실험 결과는 [1]에 수록되어 있다.

TopoX 논문에서는 TopoX 의 분할 기법과 PowerGraph 의 정점 절단 기법, PowerLyra 의 하이브리드 절단 기법을 비교하였다. 이 실험에서는 높은 메모리 사용량과 통신 비용을 이유로 PowerLyra 의 최적화되지 않은 하이브리드 절단 기법과 비교하였다. 현실 그래프에서 정점 절단 기법은 가장 높은, TopoX 의 분할 기법은 가장 낮은 복제 인자를 보여주었다. 자세한 실험 결과는 [6]에 수록되어 있다.

Gemini 논문에서는 twitter-2010 그래프에서 페이지랭크(PageRank) 수행 시간을 비교하여, 정점 절단 기법과 하이브리드 절단 기법, Gemini 의 덩어리 기반 분할 기법을 비교하였다. Gemini 시스템은 Gemini 의 덩어리 기반 분할 기법을 사용하였을 때 다른 분할 기법보다 수 배 이상 작은 질의 수행 시간을 보여주었다. 자세한 실험은 [13]에 수록되어 있다.

5. 결론

그래프 분할 기법은 그래프를 여러 분산 머신에 분할하여 저장하는 기법으로, 분산 환경에서 분산 머신 간의 통신 비용을 줄이고 부하 균형을 맞추어 그래프 질의 수행의 성능을 높이는 데 목적이 있다. 본 논문에서는 대표적인 그래프 분할 기법들과 최신 시스템의 그래프 분할 기법들을 설명하고, 기존 논문들의 실험 결과를 소개하여 비교하였다. PowerLyra 논문의 실험에서는 정점 절단 기법에 비해 최적화된 PowerLyra 의 하이브리드 절단 기법이 가장 작은 복제 인자를 보였으며, TopoX 논문의 실험에서는 TopoX 시스템의 분할 기법은 랜덤 하이브리드 기법보다 작은 복제 인자를 보였다. Gemini 논문의 실험에서는 Gemini 의 덩어리 기반 절단 기법을 사용하였을 때 가장 작은 질의 수행 시간을 보였다.

참고문헌

[1] Chen, R., Shi, J., Chen, Y., and Chen, H., "Powerlyra: Differentiated graph computation and partitioning on skewed graphs," *EuroSys*, pp. 479-494, 2015.

[2] Dubey, A., Hill, G. D., Escriva, R., Sirer, E. G., "Weaver: a high-performance, transactional graph database based on refinable timestamps," *PVLDB*, Vol. 9, No. 11, pp. 852-863, 2016.

[3] Gonzalez, J. E., Low, Y., Gu, H., Bickson, D., and Guestrin, C., "Powergraph: distributed graph-parallel computation on natural graphs," *OSDI*, pp. 17-30, 2012.

[4] Gonzalez, J. E., Xin, R. S., Dave, A., and Crankshaw, D., "GraphX: Graph Processing in a Distributed Dataflow Framework," *OSDI*, pp. 599-613, 2014.

[5] Kyrola, A., Blelloch, G., and Guestrin, C., "Graphchi: Large-scale graph computation on just a pc," *OSDI*, pp. 31-46, 2012.

[6] Li, D., Zhang, Y., Wang, J., "TopoX: topology refactorization for efficient graph partitioning and processing," *PVLDB*, Vol. 12, No. 8, pp.891-895, May 2019.

[7] Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., and Hellerstein, J. M., "Distributed GraphLab: a framework for machine learning and data mining in the cloud," *PVLDB*, Vol. 5, No. 8, pp. 716-727, 2012.

[8] Malewicz, G., Austern, M. H., Bik, A. J. C., Dehnert, J. C., Horn, I., Leiser, N., and Czajkowski, G., "Pregel: a system for large-scale graph processing," *SIGMOD*, pp. 135-146, 2010.

[9] Roy, A., Bindschaedler, L, Malicevic, J., and Zwaenepoel, W., "Chaos: Scale-out graph processing from secondary storage," *SOSP*, pp. 410-424, 2015.

[10] Shi, J., Yao, Y., Chen, R., Chen, H., Li, F., "Fast and Concurrent RDF Queries with RDMA-Based Distributed Graph Exploration," *OSDI*, pp. 317-332, 2016.

[11] Verma, S., Leslie, L. M., Shin, Y., Gupta, I., "An experimental comparison of partitioning strategies in distributed graph processing." *PVLDB*., Vol. 10, No. 5, pp. 493-504., 2017.

[12] Wang, Z., Gu, Y., Bao, Y., Yu, G., and Yu, J. X., "Hybrid pulling/pushing for i/o-efficient distributed and iterative graph computing," *SIGMOD*, pp. , 2016.

[13] Zhu, X., Chen, W., Zheng, W., and Ma, X., "Gemini: A Computation-Centric Distributed Graph Processing System," *OSDI*, pp. 301-316, 2016.