Scalable Data-driven PageRank: Algorithms, System Issues, and Lessons Learned

2014314856 소프트웨어학과 오예린

Abstract

Large Scale의 네트워크와 그래프 분석이 상당한 관심을 받고 있다. 그래프 마이닝은 반복적인 알고리즘을 자주 포함하기 때문에 더 뛰어난 퍼포먼스를 내기 위해 확인할 여러 관점이 필요하다. 이 논문에서는 PageRank를 model problem으로 하여 3가지 알고리즘 디자인 관점(work activation, data access pattern, scheduling)을 본다. 논문은 각기 다른 알고리즘 디자인의 영향을 조사한 후, 싱글 쓰레드와 멀티 쓰레드에서의 퍼포먼스에 모두 영향을 주는 이 디자인들을 이용해서 다양한 PageRank의 구현을 설계하고 테스트한다.

1. Introduction

최근의 연구에선 distributed graph analytics가 메모리를 공유함에 있어 상당한 지연을 보이고 있으며 이러한 communication cost의 증가는 전체적인 프로세싱 파워나 메모리 대역폭의 증가로도 쉽게 해결되지 않는 다는 사실이 관찰되었다. 때문에 빠른 shared-memory analytics를 구현하는 것은 cost를 감소시키고 commodity system 상에서 풍부한 applications을 가능하게 만든다. 계산이 실행되는 노드 셋을 active nodes라 정의하고 이 active nodes가 어떻게 처리되는지를 기반으로 반복되는 그래프 알고리즘을 3가지 관점에서 분류하려한다. 이 논문에서는 PageRank 알고리즘에 대한 연구를 통해 scalable data-driven graph algorithm을 설계하는 일반적인 접근방식을 제시한다. 특히, work activation, data access pattern, scheduling의 3가지 관점을 통해 PageRank의 8가지 formulations과 in-memory parallel implementations를 제시한다. data-driven formulation을 통해 active nodes를 처리하는 데 있어 알고리즘의 효율을 높이는 flexibility를 가질 수 있다.

2. Work Activation

Topology-driven과 data-driven algorithms의 work activation 관점에서 알고리즘들을 두 부류로 분류한다. topology-driven algorithm에서는 active nodes들이 그래프의 구조에 의해 정의되어 모든 노드들이 반복의 과정에서 한꺼번에 처리된다. 이에 반해 data-driven algorithm은 노드가 역동적으로 이웃노드들에 의해 활성화 된다. data-driven algorighm이 그래프에서 더 빈번이 업데이트가 일어나는 hot spots에 집중하기 때문에 topology-driven algorighm에 비해 더 효율적이다.

3. Data Access Pattern

active node가 처리될 때, 특별한 data access pattern이 보인다. 예를 들어, 어떤 알고리즘에서는 active node의 값을 읽어오고 outgoing neighbors를 업데이트 하는 반면, 다른 알고리즘은 active node의 incoming neighbors 값을 읽어온 후, active node의 값을 업데이

트한다. 따라서 이러한 data access 패턴에 의해 알고리즘을 분류하면, pull-based, pull-push-based, push-based 의 3가지의 분류가 가능하다.

1) pull-based PageRank
active node가 이웃노드의 값을 pull(read)한 후에, 자신의 값을 업데이트한다. write 연산은 active node에 대해서만 수행되기 때문에 read 연산이 write 연산에 비해 많다.

2) pull-push-based PageRank
pull-push-based 알고리즘의 경우에는, active node가 그 이웃노드의 값을 pull(read)하고, 그 이웃노드의 값을 push(update)한다. pull-push-based 알고리즘은 이웃노드에 대해 read와 write 연산을 모두 필요로 하기 때문에 pull-based 알고리즘에 비해 cost가 높다. 그러나 active node가 수동적으로 이웃노드들로부터 정보를 받는 pull-based에 비해 pull-push-based는 active node가 이웃노드로 정보를 전달할 수 있다는 점에서는 이점이다.

3) push-based PageRank
active node가 자신의 값을 업데이트하고, 이웃노드의 값을 push(update)한다. pull-based 알고리즘과 비교했을 때, write 횟수가 더 많기 때문에 cost가 더 높다. 하지만, 빈번한 업데이트가 있기 때문에 네트워크 전반에 정보를 빠르게 전달할 수 있다. read/write 연산을 모두 하는 pull-push-based 알고리즘과 비교하면, push-based algorithm은 write 연산만 하기 때문에 더 효율적이다.

4. Scheduling
어떤 task의 순서로 실행될지 정하는 Task scheduling은 그래프 알고리즘에서 매우 중요하다. 예를 들어, data-driven PageRank의 경우에는 residual r_v를 갖고 있는 노드의 PageRank가 업데이트 되면, total residual은 $r_v(1-a)$로 감소하게 된다. 이 사실은 large residual node를 먼저 처리하게 되면, 알고리즘이 빠르게 수렴할 수 있음을 말한다. priority scheduling에서는 각 task가 priority, value가 증가 또는 감소하는 순서로 스케줄된다. 이 논문에서는 priority scheduler의 다른 디자인에서 PageRank 의 sensitivity를 알기위해 두 가지 종류의 디자인을 이용했다. 하나는 set-semantics 대신에 priority fidelity를 선호하는 것이고, 다른 하나는 priority fidelity 대신에 set-semantics를 보존하는 것이다. 논문에서 이용한 첫 번째 scheduler는 scalable한, NUMA-aware OBIM priority scheduler이다. 두 번째로 이용한 scheduler는 bulk-synchronous priority scheduler이다.

5. Experimental Results
PageRank의 성능과 scaling sensitivity를 보기 위해, 다른 종류의 스케줄링과 data access pattern을 적용하며 다양한 PageRank 알고리즘들을 구현했다. 데이터로는 Twitter, Friendster, pld, sd1과 같은 소셜 네트워크와 하이퍼링크 그래프를 이용했다. data-driven implementations이 다른 반복 methods들보다(topology implementaion) 월등히 빨랐다. 또한, push-only implementations이 pull-push보다 성능이 뛰어났으며 priority-scheduler (노드가 새로운 priority bin에 들어갈 때마다 중복된 task를 포함해야 해서 많은 tasks가 useless) non-priority scheduler보다 scale은 잘 했으나 성능은 좋지 않았다.