SIMD 최적화를 이용한 CPU 기반 그래프 엔진의 성능 개선

조익현 [1], 장명환 [2], 김상욱 [3*]
[1] 한양대학교 컴퓨터·소프트웨어학과 석사과정
[2] 한양대학교 컴퓨터·소프트웨어학과 박사과정
[3] 한양대학교 컴퓨터·소프트웨어학과 교수

childyouth@hanyang.ac.kr, sugichiin@hanyang.ac.kr, wook@hanyang.ac.kr

SIMD Optimization for Improving the Performance of a CPU-Based Graph Engine

Ikhyeon Jo, Myung-Hwan Jang, Sang-Wook Kim*
Department of Computer and Software, Hanyang University

요 약

Single-machine-based 그래프 엔진의 state-of-the-art 모델인 RealGraph 는 쓰레드를 이용한 병렬화로 성능을 향상하였으나 쓰레드 내부에서의 병렬성은 고려되지 않았다. 본 논문은 SIMD 명령어를 이용해 RealGraph 의 병렬성을 향상시켰다. 쓰레드 내부의 효율성을 높이기 위해 RealGraph 의 구조와 그래프 알고리즘의 분석을 통한 SIMD 명령어의 적용 가능한 영역을 탐색하였다. 실험으로 SIMD 명령어의 적용을 통해 쓰레드 내부에서 벡터 연산을 수행하여 평균 7.6%, 11.7%, 9.2%의 수행 시간 단축을 이끌어냈으며 SIMD 명령어의 적용이 그래프 엔진의 분석 성능에 얼마나 도움이 될 수 있는지 확인하였다.

1. 서론

최근 실 세계 네트워크를 이용한 분석이 활발히 진행됨에 따라 SNS, 웹 등 실 세계 네트워크의 객체 간 상호 관계를 표현하는 그래프 자료구조를 통한 분석 기술들이 활발하게 연구되고 있다. 그래프는 네트워크 객체 및 객체 간의 관계를 각각 정점과 간선으로 표현하며 네트워크에 내재되어 있는 유용한 지식을 추출해 내기 위해 다양한 그래프 알고리즘이 사용되고 있다 [1, 2, 3]. 실 세계 그래프의 크기가 커져감에 따라 빠르고 효율적인 그래프 분석을 위해 그래프 엔진에 대한 연구가 진행되어 왔다 [4, 5, 6, 7, 8, 9].

Single-machine-based 그래프 엔진 [4, 5, 6, 7] 은 전체 그래프를 단일 머신이 보유한 자원만으로 그래프를 분석하는 방식이다. 그 중에서도 메인 메모리에서 수용 불가능한 대규모 그래프를 처리하기 위해 디스크를 활용하는 디스크 기반 분석 방식은 많은 수의 머신을 활용하여 분석하는 distributed-system-based 그래프 엔진 [8, 9] 과 유사한 성능을 저렴한 비용으로 달성하였다. Single-machine-based 그래프 엔진은 한정된 자원으로 최선의 성능을 이끌어 내기 위해 IO 를 최적화하거나 효율적인 병렬 처리를 위한 작업량 분배 등 다양한 최적화 방식을 적용하고 있다.

본 논문에서는 기존의 CPU 기반 single-machine-based 그래프 엔진의 state-of-the-art 모델인 RealGraph [4] 에서 프로세싱 영역에서의 추가 최적화가 가능한 영역에 대해 특정하고 CPU 성능 개선을 위해 병렬 수행성을 향상시킬 수 있도록 SIMD (Single-Instruction-Multiple-Data) 기반 명령어를 적용하여 그래프 엔진의 성능향상을 측정한다.

2. 관련 연구

REALGRAPH. RealGraph 는 실 세계 그래프가 가진 power-law 분포 특징을 이용해 효과적으로 대규모 실 세계 그래프 분석을 수행하는 single-machine-based

* 교신 저자

그래프 엔진의 state-of-the-art 모델이다. RealGraph 는 4 개의 계층 – 각각 storage, buffer, object, CPU thread management 으로 구성되어 있으며 이러한 아키텍처는 외부 저장소(e.g., HDD 혹은 SSD)에 저장되어 있던 그래프 데이터를 메인 메모리로 빠르게 읽어와 쓰레드에 분배하고 계산하는 일련의 그래프 알고리즘 계산 과정을 최적화 할 수 있도록 설계되었다.

그래프 데이터의 저장을 위해 RealGraph 는 1MB 의 블록에 정점 ID 와 인접리스트 쌍을 저장하게 된다. 인접리스트의 크기는 정점 마다 크게 차이가 나기 때문에 블록마다 하나의 정점 쌍을 저장한다면 저장공간을 낭비하거나 인접리스트 전체를 저장할 수 없게 된다. RealGraph 에서는 간선이 집중적으로 연결되어 있어 인접리스트가 블록 내부에 전부 저장할 수 없는 정점 쌍 에게 여러 개의 블록에 걸쳐 저장하도록 유도하고 이 블록들을 large block 이라 한다. Large block 에 속하지 않는 정점 쌍들에 대해서는 같이 저장되도록 하며 이렇게 저장된 블록들은 small block 으로 구분한다. 실 세계 그래프의 power-law 분포 특징 [10] 으로 인해 블록은 소수의 large block 과 대다수의 small block 으로 이루어진다. Large block 과 small block 는 정점 쌍이 겹치지 않도록 저장된다. RealGraph 는 블록단위로 각 쓰레드에 할당하고 블록에 저장된 정점과 인접리스트를 순차적으로 처리한다. large block 과 small block 에 저장된 데이터의 크기가 크게 다르지 않기 때문에 균등한 분배가 가능하다.

SIMD. 기본적으로 CPU 는 내부에서 하나의 데이터에 대해 하나의 명령어를 수행하는 SISD (Single-Instruction-Single-Data) 기법으로 동작한다. 이러한 SISD 기법은 동일 명령어를 반복 수행하는데 있어 효율성이 낮다. SIMD(Single-Instruction-Multiple-Data)는 명령어를 동시에 수행하기 위한 기술로, 데이터들을 벡터로 만들어 레지스터에 적재하고 명령어 하나로 벡터를 처리하도록 만든다. 빠른 병렬 처리가 필요한 그래프 엔진 특성 상 SIMD 를 이용한 최적화가 연구되어 왔다 [7].

3. 제안 방법

RealGraph 는 그래프를 블록으로 분할하고 쓰레드에 분산시켜 처리함으로서 효율적인 병렬 처리를 이루어내고 있다. 하지만 large block 을 기준으로 1MB 블록 내부에는 수만에서 수십만 개의 데이터가 저장되어 있으며 이러한 거대한 크기는 블록 내부의 병렬 처리 또한 필요함을 시사한다. 우리는 블록 내부에서 정점과 인접리스트 각각을 순차적으로 접근하여 처리한다는 점을 들어 SIMD 명령어를 통한 그래프 엔진의 최적화를 생각했다. RealGraph 의 특징과 그래프 알고리즘 코드 각각에서 SIMD 명령어를 적용할 수 있는 부분에 대해 탐색했다.

RealGraph 의 SIMD applicability. RealGraph 는 그래프 알고리즘에 필요한 속성값을 정점 개수만큼의 길이를 갖는 벡터로 관리한다. RealGraph 는 블록 내부의 정점에 대해, 그리고 해당 정점의 인접리스트에 대해서 순차접근하여 필요한 속성값을 찾게 된다. 이 과정에서 정점 ID 에 대한 속성값 벡터의 offset 을 계산하는 일련의 계산이 순차적으로 반복된다.

그래프 알고리즘의 SIMD applicability. 그래프 알고리즘은 간선으로 연결된 정점의 속성값을 취합하거나 비교하는 계산을 수행하게 된다. 이러한 계산은 인접리스트를 연속적으로 접근하고 알고리즘 계산에 필요한 명령어를 동일 반복 수행하는 과정으로 이루어진다.

본 논문에서는 위의 두 가지 영역에서 SIMD 의 적용 가능성을 확인하였다. 이에 따라 각 인접리스트를 일정 길이의 벡터 단위로 접근하도록 변경하고 SISD 기법으로 동작되던 기존의 코드를 수정하였다. 결과적으로 인접리스트를 벡터 단위로 순차접근하고, 해당 벡터들에 대해 SIMD 명령어들을 사용하여 쓰레드 내부에서 병렬수행성을 향상시켰다.

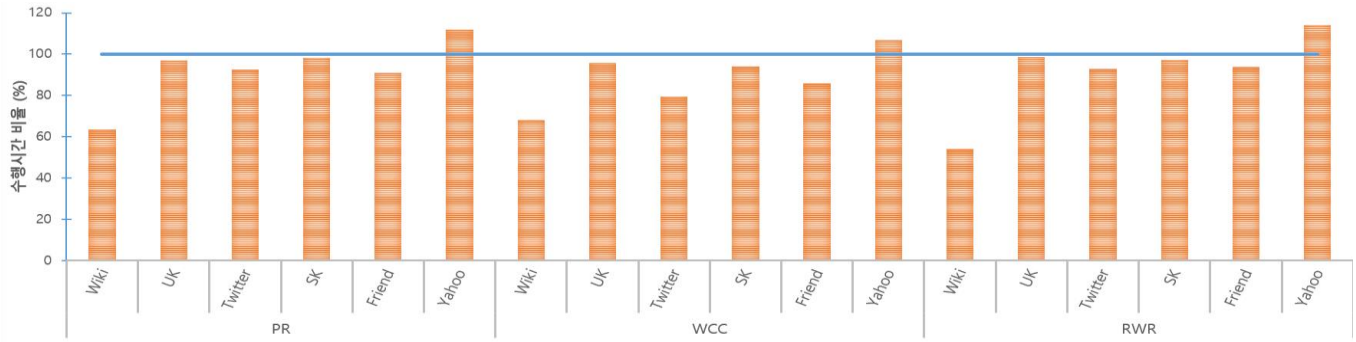
<표 1> 실 세계 그래프 데이터 정보

데이터	Wiki	UK	Twitter	SK	Friend	Yahoo
# 정점	12M	39M	61M	50M	68M	1.4B
# 간선	370M	930M	1.4B	1.9B	2.5B	6.6B
저장 크기	5.7GB	16GB	24GB	32GB	44GB	114GB
# 블록	1559	3957	5954	8410	10393	32737
Avg. 간선 (정점)	31.12	23.73	35.25	38.49	37.83	9.21
Avg. 간선 (small block)	243,290	235,629	246,734	232,859	248,835	203,112

4. 실험

본 논문은 state-of-the-art 그래프 엔진인 RealGraph 에서 SIMD 최적화를 적용하기 위해 Intel 의 AVX2 [11] 라이브러리를 사용했다. 속성값 벡터의 offset 계산과 그래프 알고리즘의 수행을 위해 set, load, sub, gather, add, hadd 와 같은 연산을 AVX 명령어로 대체하였다. 우리는 Xeon 4309Y CPU 2 개 와 SSD 1 개 가 장착된 R750 단일 머신에서 실험을 수행했다. 비교를 위해 6 개의 실 세계 그래프 데이터 [4] <표 1> 에서 총 세 가지 알고리즘 – 각각 Page-Rank(PR), Weakly Connected Components(WCC), Random Walk with Restart(RWR) [1, 2, 3] 을 수행했다. RealGraph 의 Hyper-Parameter 는 각각 CPU 쓰레드를 8, 메모리 제한을 16GB 로 설정하였다.

(그림 1)은 SIMD 를 적용하여 최적화한 RealGraph (RG_SIMD)의 수행 시간을 측정한 결과로 Y 축은 RG_SIMD 와 베이스라인인 기존의 RealGraph(RG_OG) 의 수행 시간 비율을 나타낸다. 수치가 낮을수록 수행 시간을 줄였음을 의미하며 100%는 RG_SIMD 의 수행 시간이 RG_OG 와 동일함을 의미한다. 대부분의 결과에서 RG_OG 에 비해 RG_SIMD 가 수행 시간이 줄어들었음을 알 수 있다. 그래프 데이터 별 성능향



(그림 1) 기존 RealGraph 수행 시간(100%)에 대한 SIMD 최적화의 수행 시간 비율

상의 차이가 존재하는데, 이는 그래프 데이터가 저장된 블록의 특징으로 인해 발생한다. 앞서 언급했듯이 실 세계 그래프는 power-law 분포 특성 때문에 소수의 large block 과 다수의 small block 으로 구성된다. 대다수의 정점을 차지하는 small block 들에서 SIMD 를 통한 최적화를 극대화하기 위해서는 small block 의 인접리스트들의 평균 길이(즉, 정점의 평균 간선 개수)가 길어야 한다. AVX2 의 최적화를 위해서는 길이가 8 인 벡터를 두 개 사용하기 때문에 인접리스트의 길이가 최소 16 이 되어야 한다. 하지만 <표 1>에서 볼 수 있듯이 Yahoo 그래프 데이터의 정점 별 평균 간선 개수는 9.21 개로 AVX2 를 통한 최적화를 기대하기 힘들며, 오히려 AVX2 를 수행 할 수 있는지 판단하는 과정이 오버헤드가 되어 수행 시간이 소폭 증가한 것으로 보여진다. 또한 small block 별 평균 간선 개수가 상대적으로 적은 Yahoo, SK, UK 데이터와 Friend, Twitter, Wiki 사이의 성능 향상 폭의 차이를 통해 정점 별 간선 개수가 SIMD 최적화에 미치는 영향을 확인할 수 있다.

결과적으로 RG_SIMD 는 평균적으로 9.22%의 성능 향상을 보여주었으며 PR, WCC, RWR 에 대해 각각 최대 36.4%, 31.9%, 46.1%의 성능향상을, 평균 7.6%, 11.7%, 9.2%의 성능향상을 이루었다.

5. 결론

본 논문에서는 쓰레드를 통한 병렬최적화가 되어있는 state-of-the-art 그래프 엔진인 RealGraph 에서 SIMD 명령어를 통해 추가적인 병렬화를 통한 성능향상을 제시했다. 우리는 RealGraph 의 설계적 특징과 그래프 알고리즘의 특징을 파악해 offset 의 계산, 정점 값 취합 영역에서 SIMD 명령어를 통한 최적화가 가능함을 확인했다. 실험을 통해 SIMD 명령어가 실 세계 그래프 분석 성능을 향상시킬 수 있다는 가능성을 확인하였다.

사사

본 연구는 삼성전자와 한양대의 산학 협력의 일환으로 수행되었음. 또한 본 연구는 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행되었음 (No.2022-0-00352 및 No.RS-2022-00155586 (실세계의 다양한 다운스트림 태스크를 위한 고성능 빅 하이퍼그래프 마이닝 플랫폼 개발, SW 스타랩)).

참고문헌

[1] Lawrence Page, et al., “The pagerank citation ranking: Bring order to the web.", technical report, Stanford University, 1998.

[2] Kenji Suzuki, Isao Horiba, and Noboru Sugie, "Linear-time connected-component labeling based on sequential local operations.", Computer Vision and Image Understanding, 89, 1, 1-23, 2003.

[3] Hilmi Yildirim, and Mukkai S. Krishnamoorthy, "A random walk method for alleviating the sparsity problem in collaborative filtering.", Proceedings of the 2008 ACM conference on Recommender systems, 2008.

[4] Yong-Yeon Jo, et al., "Realgraph: A graph engine leveraging the power-law distribution of real-world graphs.", The World Wide Web Conference, 2019.

[5] Aapo Kyrola, Guy Blelloch, and Carlos Guestrin, "Graphchi: Large-scale graph computation on just a PC.", Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12), 2012.

[6] Amitabha Roy, Ivo Mihailovic, and Willy Zwaenepoel, "X-stream: Edge-centric graph processing using streaming partitions.", Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, 2013.

[7] Shuo Han, et al., "Speeding up set intersections in graph algorithms using simd instructions.", Proceedings of the 2018 International Conference on Management of Data, 2018.

[8] Joseph E. Gonzalez, et al., "Powergraph: Distributed graph-parallel computation on natural graphs.", Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12), 2012.

[9] Joseph E. Gonzalez, et al., "Graphx: Graph processing in a distributed dataflow framework.", 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), 2014.

[10] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos, "Graph evolution: Densification and shrinking diameters.", ACM transactions on Knowledge Discovery from Data (TKDD), 1, 1, 2-es, 2007.

[11] Chris. Lomont, "Introduction to intel advanced vector extensions.", Intel white paper, 23, 2011.