단일 PC기반 그래프 처리 가속화를 위한 저장장치 내부 응용 관리 캐시

천명준○, 한승욱, 김지홍
서울대학교 컴퓨터공학부
{mjchun, swhan, jihong}@davinci.snu.ac.kr

In-Storage, Software-Managed Cache for Accelerating Graph Processing on a Single PC

Myoungjun Chun○, Seungwook Han, Jihong Kim
Department of Computer Science and Engineering, Seoul National University

요 약

단일 PC 기반 그래프 처리는 여러 노드 간 관리 비용 없이 대규모 그래프를 처리할 수 있지만, 빈번한 입출력으로 인한 지연이 발생한다. 본 논문에서는 빈번한 입출력이 반복되는 부분 그래프 읽기와 데이터 접근의 낮은 지역성으로 인해 발생함을 보이고, 이를 극복할 수 있는 저장장치 내부 응용 관리 캐시를 제안한다. 저장장치 내부 응용 관리 캐시는 저장장치의 디램 공간을 사용하며, 응용에 의해 직접적으로 관리된다. 새로 제안하는 캐시에서는 응용이 추후 읽을 부분 그래프를 예측하여 캐시에 미리 적재함으로써, 읽기 지연을 줄이고 캐시 적중률을 높일 수 있다. 실험 결과, 기존 기법 대비 평균 20%의 성능 향상을 확인하였다.

1. 서 론

최근 여러 소셜 네트워크와 웹 그래프가 널리 사용됨에 따라 그래프 처리 연구의 중요성이 커지고 있다. 분산 그래프 처리 시스템[1,2]들은 대규모 그래프를 처리할 수 있으나 여러 노드 간 관리에 높은 비용을 요구하는 문제가 있다. 때문에, 단일 PC 기반에서 대규모 그래프를 처리하기 위한 연구들이 제안되었다[3,4].

단일 PC기반 그래프 처리는 대규모 그래프를 호스트 메모리에 들어갈 수 있는 크기의 부분 그래프(Subgraph)들로 나누어져 수행된다. 부분 그래프들은 저장장치에 저장되어 있기 때문에, 처리되기 전 저장장치에서 호스트 메모리로 적재되고 처리된 후 다시 저장장치에 쓰여진다. 다시 말해서, 단일 PC기반 그래프 처리의 전체 처리 시간은 읽기/쓰기 시간과 그래프 처리 시간으로 나눌 수 있다. 그래프 처리 시간에 저장장치는 유휴 상태(Idle state)이지만 처리되고 있는 부분 그래프가 호스트 메모리를 모두 점유하고 있어, 다음 부분 그래프 읽기는 현재 부분 그래프 처리가 끝나야 수행될 수 있다. 따라서 전체 처리 시간의 많은 부분이 부분 그래프 읽기에 소요되는 문제가 있다.

단일 PC기반 그래프 처리 시 데이터 접근 패턴이 높은 지역성을 가지고 있을 경우 부분 그래프 읽기가 수행될 때 시스템의 여러 계층의 캐시에서 대부분이 적중하게 되어, 읽기 시간에 저장장치의 느린 지연 시간이 드러나지 않는다. 하지만 단일 PC기반 그래프 처리 시 데이터 접근 패턴은 공통된 부분이 없는 분할된 부분 그래프들이 순차적으로 접근되는 패턴이며, 전체 데이터에 해당하는 대규모 그래프의 크기가 캐시의 크기보다 훨씬 크다. 따라서 LRU 정책을 기반으로 동작하는 기존 캐시로는 순차 범람(Sequential flooding) 현상이 일어나게 되어 대부분 접근에서 캐시 실패(Cache miss)가 발생하고, 낮은 캐시 적중률(Hit ratio)을 보인다.

본 논문에서는 단일 PC기반 그래프 처리의 위 두 가지 문제점을 해결하기 위한, 저장장치 내부 응용 관리(Software-managed) 캐시를 제안한다. 이 새로운 캐시는 SSD (Solid State Drive)와 같은 저장장치 내부의 디램 공간을 사용하며 응용이 유저 레벨의 라이브러리를 통해 데이터를 캐시에 미리 읽기(Prefetch)하도록 지시할 수 있다. 단일 PC기반 그래프 처리 응용이 그래프 처리 시간에 추후 접근될 부분 그래프를 예측하여 저장장치가 내부 디램 캐시에 미리 읽기 하도록 지시함으로써 높은 캐시 적중률을 달성하고, 읽기 시간을 감소시켜 총 처리를 가속화하는 것이 본 연구의 목표이다.

기법의 유효성을 검증하기 위해 SSD 에뮬레이터[5]에서 저장장치 내부 응용 관리 캐시를 구현한 뒤 단일 PC기반 그래프 처리 응용인 GraphChi[3]에서의 성능 개선을 평가한 결과, 본 기법을 사용하지 않았을 때와 비교하여 전체 성능이 평균 20% 개선됨을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 대표적인 단일 PC기반 그래프 처리 응용인 GraphChi의 동작 과정에 대해 설명하고, 데이터 접근 패턴을 분석한다. 3장에서는 저장장치 내부 응용 관리 캐시의 설계 및 구현과 GraphChi에 적용되었을 때의 동작을 설명한다. 이어 4장에서는 GraphChi에 본 기법을 적용했을 때의 성능 개선을 SSD 에뮬레이터[5] 환경에서 평가한 결과를 보이고, 5장에서 결론을 맺는다.

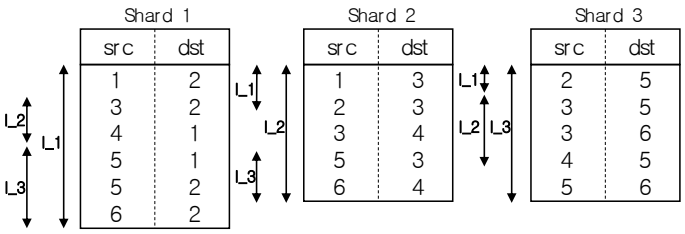

그림 1. GraphChi 데이터 접근 패턴 예시

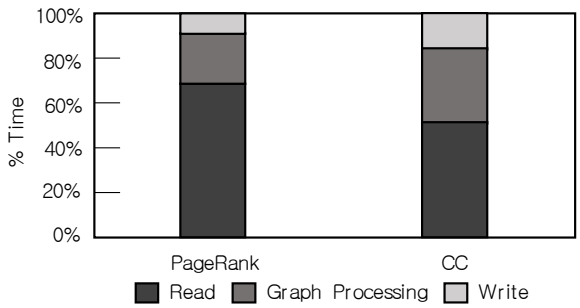

그림 2. 읽기 시간, 그래프 처리 시간, 쓰기 시간의 비율

2. GraphChi 분석

2.1 동작 과정

GraphChi는 그래프를 처리하기 전 대규모 그래프를 정점(Vertex)들의 부분 집합인 interval 단위로 분할한다. 하나의 interval에 해당하는 정점들의 진입 간선(In-edge)들은 전 처리 과정에 정점에 따라 정렬되어 하나의 shard를 구성한다. 부분 그래프는 각 interval에 해당하는 정점들과 연관된 간선으로 구성되는데, 부분 그래프의 크기는 가용 가능한 호스트 메모리의 최대 크기에 따라 결정된다. 즉, 그래프의 크기가 커지더라도 각 부분 그래프의 크기가 커지는 것이 아니라 수행해야 하는 interval의 횟수가 증가하게 된다.

전 처리 과정이 끝나면 순차적으로 각 interval에 대한 그래프 처리가 수행된다. 첫 번째로, 현재 interval에 해당하는 shard에서 진입 간선들을 읽고 퍼져 있는 다른 shard들에서 진출 간선(Out-edge)들을 읽어 부분 그래프를 구성한다. 두 번째 단계는, 부분 그래프를 사용자가 정의한 함수에 따라 처리하는 단계이며 병렬적으로 빠르게 처리될 수 있다. 마지막으로, 바뀐 결과를 저장장치에 다시 쓴다. 이 세 단계가 전체 interval에 대해 수행되면 전체 그래프 처리가 끝난다.

2.2 데이터 접근 패턴

그림 1은 GraphChi의 데이터 접근 패턴의 예시를 보여준다. I_n은 n번째 interval에서 접근해야 하는 데이터를 나타낸다. Shard의 간선들은 전 처리 과정에서 정점에 따라 정렬되어 있기 때문에 n번째 interval에서 접근해야 하는 데이터들은 예측 가능하다. 그림 2는 읽기/쓰기 시간과 그래프 처리 시간이 전체 처리 시간에서 차지하는 비율을 보여준다. 그림2에서 확인할 수 있듯이, 전체 처리 시간의 60%가 부분 그래프 읽기 시간에 소요되고 있어, 읽기 시간을 줄이면 전체 처리 시간을 효과적으로 줄일 수 있다.

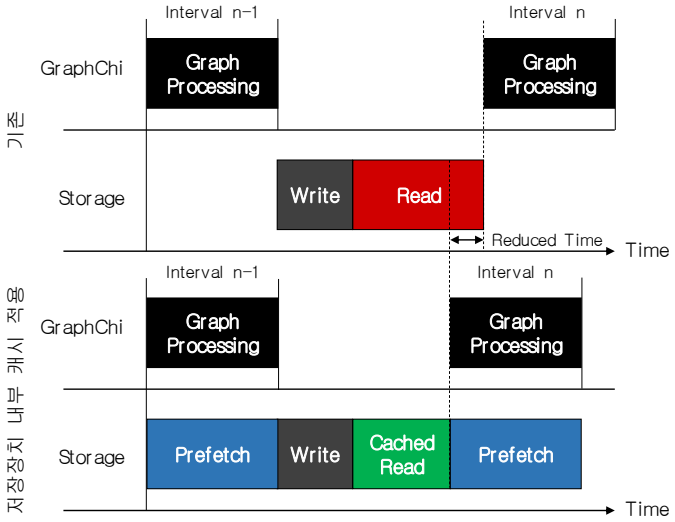

그림 3. 저장장치 내부 응용 관리 캐시 적용 유무에 따른 GraphChi 동작 과정

3. In-Storage, Software-Managed Cache

3.1 설계 및 구현

저장장치 내부 응용 관리 캐시는 SSD(Solid State Drive)나 HDD(Hard Disk Drive) 내부 디램의 일부 영역이다. 이 영역을 응용 레벨에서 직접 관리하기 위해서는 응용이 호출하여 사용할 수 있는 유저 레벨 라이브러리가 필수적이다. 응용이 유저 레벨 라이브러리를 호출하면, 유저 레벨 라이브러리는 ioctl 명령을 저장장치에 보낸다. Ioctl 명령이 저장장치에 전달되면, 저장장치의 펌웨어가 명령에 따른 작업을 수행한다.

유저 레벨 라이브러리가 제공하는 세 가지의 인터페이스 함수는 다음과 같다. (1) *prefetch* 함수는 인자로 파일 이름을 전달 받아, 대상 파일을 구성하는 저장장치의 물리 페이지(Physical page)들을 디램 캐시에 적재한다. 이 동작은 비동기적으로 진행되기 때문에 함수는 ioctl 명령 후 곧바로 반환되어 응용이 다른 작업을 진행할 수 있다. (2) *is_prefetch* 함수는 인자로 전달받은 파일의 모든 물리 페이지들이 디램 캐시에 적재되어 있는지를 확인한다. 확인 작업이 모두 완료되면 결과가 ioctl의 반환 값으로 응용에 전달된다. (3) *set_victim* 함수는 대상 파일을 구성하는 물리 페이지들을 희생양(Victim)으로 선정한다. 희생양으로 선정된 물리 페이지들은 캐시에 여유 공간이 없어 캐시 교체가 일어날 시 가장 우선적으로 교체된다. 캐시 공간이 충분하다면 희생양으로 선정되어도 캐시 교체가 일어나지 않기 때문에 바로 교체되는 것은 아니다.

현재 상용화된 저장장치의 펌웨어는 위의 기능들을 수행할 수 있는 인터페이스를 지원하지 않기 때문에, 본 연구에서는 SSD 에뮬레이터[5]를 수정하여 저장장치 내부 응용 관리 캐시를 구현하였다.

3.2 적 용

저장장치 내부 응용 관리 캐시를 적용하기 위해, GraphChi가 유저 레벨 라이브러리 함수를 알맞게 호출하도록

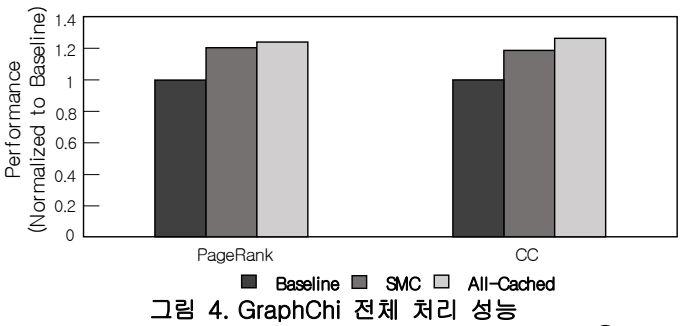

그림 4. GraphChi 전체 처리 성능

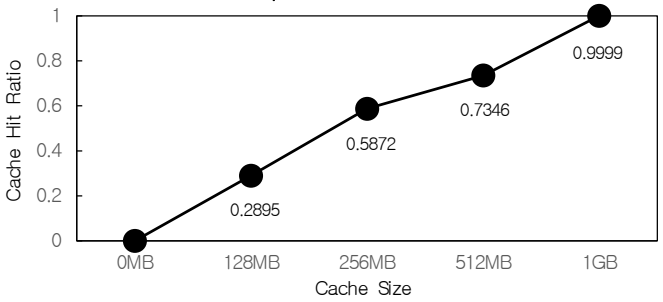

그림 5. 캐시 크기에 따른 캐시 적중률 변화

수정하였다. 그림 1에서 GraphChi가 n번째 interval에서 접근해야 하는 데이터들은 예측 가능함을 보였으므로, n-1번째 interval의 그래프 처리 시간에 n번째 interval에서 접근할 shard 파일을 인자로 *prefetch* 함수를 호출한다. 이후 n번째 interval의 부분 그래프 읽기는 연관된 shard 파일이 캐시에 적재되어 있으므로 빨리 처리될 수 있다. n번째 interval의 그래프 처리가 끝나면, 연관된 shard 파일을 *set_victim* 함수를 호출하여 캐시 교체가 일어날 때 우선적으로 교체되도록 한다. 그림 3은 저장장치 내부 응용 관리 캐시 적용 유무에 따른 GraphChi의 동작 과정을 비교하여 보여준다. 그림 3에서 볼 수 있듯, 기존 GraphChi에서는 그래프 처리 시간에 저장장치가 작업을 하지 않지만, 저장장치 내부 응용 관리 캐시를 사용하면 추후 접근될 데이터를 저장장치가 캐시에 미리 읽기 할 수 있다. 결과적으로, 부분 그래프 읽기가 캐시에서 이루어져 읽기 시간이 감소한다. 그림 2에서 읽기 시간이 전체 처리 시간에서 높은 비율을 차지하는 것을 보았기 때문에, 읽기 시간 감소가 GraphChi의 성능을 크게 개선시킬 것으로 기대할 수 있다.

4. 실 험

4.1 실험 환경

모든 실험은 Intel i7-4790 3.6Ghz 8-스레드 CPU, PC3-10600 16GB 메인 메모리, Linux 4.4.0-38 환경에서 SSD 에뮬레이터[5]를 사용하여 진행하였다. 에뮬레이터는 저장장치 내부 응용 관리 캐시를 지원하도록 수정하였다. 메인 메모리의 10GB 가량은 에뮬레이터를 위한 공간으로, 4GB 가량은 GraphChi를 위한 공간으로 할당하였다. 대상 그래프 데이터는 2,601,977개의 정점과 63,497,050개의 간선을 가진 소셜 그래프 sx-stackoverflow를 사용하였다.

4.2 실험 결과 및 분석

저장장치 내부 응용 캐시의 유효성을 평가하기 위해, 구현한 에뮬레이터에서 대상 그래프 데이터를 PageRank, ConnectedComponents (CC) 두 알고리즘으로 GraphChi를 통해 처리했을 때의 전체 처리 성능을 측정하였다. 그림 4는 기존 GraphChi (**Baseline**), 1GB 크기의 저장 장치 내부 응용 캐시를 적용한 GraphChi (**SMC**), 모든 데이터가 미리 페이지 캐시에 적재되어 있는 GraphChi (**All-Cached**)의 전체 처리 성능을 Baseline을 기준으로 정규화(Normalize)하여 나타낸 것이다. 그림 4에서 확인할 수 있듯, 제안하는 저장 장치 내부 응용 캐시를 통해 GraphChi의 전체 처리 시간이 평균 20% 개선되었다. 추가적으로, 모든 데이터가 페이지 캐시에 미리 적재되어 있을 때의 성능에 4.6% 근접하는 것을 확인하였다. 이 결과를 통해, 저장 장치 내부 응용 캐시의 적용이 단일 PC기반 그래프 처리의 낮은 시스템 자체 캐시 적중률 문제의 해법이 될 수 있을 것으로 예상할 수 있다.

그림 5는 저장장치 내부 응용 관리 캐시의 크기를 변화시키며 GraphChi를 수행했을 때 읽기 데이터 접근이 캐시에서 적중한 비율을 나타낸다. 그림 5에서 볼 수 있듯, 저장장치 내부 응용 관리 캐시의 크기에 따라 캐시 적중률이 증가하며, 1GB 크기일 때 99% 캐시 적중률을 보인다. 저장장치 내부 응용 관리 캐시는 GraphChi가 데이터 패턴을 예측하여 직접 관리하기 때문에, 호스트 메모리의 25%인 1GB 캐시 크기로 높은 캐시 적중률을 달성할 수 있었다.

5. 결 론

본 논문에서는 단일 PC기반 그래프 처리 응용이 저장장치 내부의 디램 공간 일부를 직접 관리하는 저장장치 내부 응용 관리 캐시를 제안하였다. 저장장치 내부 응용 관리 캐시를 통해, 단일 PC기반 그래프 처리 응용이 그래프 처리 시간에 추후 접근할 데이터를 미리 읽기 함으로써 읽기 시간을 줄이고, 높은 캐시 적중률을 달성할 수 있었다. 실험 결과, 단일 PC기반 그래프 처리 응용의 전체 성능이 20% 향상됨을 확인하였다.

감사의 글

이 연구를 위해 연구장비를 지원하고 공간을 제공한 서울대학교 컴퓨터연구소에 감사드립니다. 이 논문은 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2015M3C4A7065645). (교신저자: 김지홍)

참 고 문 헌

[1] G. Malewicz et al., “Pregel: A System for Large-Scale Graph Processing,” *SIGMOD*, pp. 135–146, 2010.

[2] J. E. Gonzalez et al., “GraphX: Graph Processing in a Distributed Dataflow Framework,” *OSDI*, pp. 599-613, 2014.

[3] A. Kyrola et al., “GraphChi: Large-scale Graph Computation on Just a PC,” *OSDI*, pp. 31-46, 2012.

[4] X. Zhu et al., “GridGraph: Large-Scale Graph Processing on a Single Machine Using 2-Level Hierarchical Partitioning,” *USENIX ATC*, pp. 375-386, 2015.

[5] J. Sang-Woo et al., “BlueDBM: An Appliance for Big Data Analytics,” *ISCA*, pp. 1-13, 2015.