이기종 저장장치를 위한 제거 비용 평가 기반 효과적인 캐시 관리 기법

(An efficient cache management based on eviction cost estimation for heterogeneous storage devices)

박 세 진*, 박 찬 익
포항공과대학교 컴퓨터공학과
(Sejin Park*, Chanik Park)
(Department of Computer Science and Engineering, POSTECH)

Abstract : The objective of cache is to reduce I/O access of physical storage device so that user accesses their data faster. Traditionally, the most important metric to measure the performance of cache is hitratio. Thus, when the cache maintains hitratio high, it is regarded as a good cache replacement policy. However, the cache miss latency is different when the storages are heterogeneous. Though the cache hitratio is high, if the cache often misses with low performance disk, then the user experiences low performance. To address this problem we proposed eviction cost estimation based cache management. In our result, the eviction cost estimation based cache management has 10~30% throughput improvement compared with LRU cache management.

Keywords : cache heterogeneous storage eviction estimation

I. 서 론

로컬에 있는 저장장치는 다양한 저장장치와 [1,2,3] 혼용이 되어 많이 쓰인다. 기존의 네트워크 기반 저장장치들을 비롯해, 최근에 많이 나오는 클라우드 기반 스토리지 서비스[4], 그리고 Solid State Drive (SSD) 등의 고성능 저장장치들이 많이 보급 되고, 이러한 저장장치는 기존의 하드 디스크와 혼용되어 사용되고 있다.

일반적으로 성능이 다른 저장장치가 있을 경우, 자주 사용하는 데이터를 고 성능 저장장치에 배치시키는 것이 유리한데, HDD와 SSD가 혼용되어 사용되는 시스템을 예로 들면 데이터 재배치에 크게 다음 두 가지의 방법이 있다.

1. 사용자가 데이터 직접 접근 빈도수를 예상하여 수작업으로 HDD 와 SSD에 데이터를 재배치시키는 방법
2. 데이터 재배치 소프트웨어가 사용자의 데이터 사용빈도를 측정 후 자동으로 배치시키는 방법

1번 의 경우, 사용자가 직접 접근 빈도수를 측정하기가 쉽지 않으며, 저장장치에 대한 지식이 없는 사람들은 직접 할 수 없다. 2번의 경우, 사용자의 의도와 다르게 데이터들의 재배치가 일어나게 되어, 추후 데이터 검색에 혼란을 야기될 수 있다.

본 논문에서는 이러한 이기종 저장장치의 특성을 고려한 고성능 캐시 정책을 제안한다. 제안하는 캐시를 사용하면 추가적인 데이터 재배치 없이, 고성능 효과를 낼 수 있게 된다. 2장에서는 기반 지식을 설명하며 3장에서는 설계를 설명하고, 4장에서

* 교신저자(Corresponding Author)
박찬익 : 포항공과대학교 컴퓨터공학과
박세진 : 포항공과대학교 컴퓨터공학과
※ 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2011-0016972)

는 제안하는 캐시 성능 평가를 한다. 그리고 5장에서 결론을 맺으며 본 논문을 정리한다.

Ⅱ. 기반 지식

캐시는 저장장치에 접근하는 입출력 횟수를 줄여서, 사용자가 더 빠르게 데이터에 접근 할 수 있게 해 준다. 전체 시스템의 병목현상을 일으키는 저장장치 접근을 효율적으로 처리하는 캐시는 서버, 데스크탑 시스템 뿐 만 아니라, 임베디드 시스템에서도 매우 중요하다. 이러한 캐시의 성능은 캐시 적중률을 통해 대변되는데, 일반적으로 다양한 워크로드에 대해서 높은 캐시 적중률을 유지시킬 수 있는 캐시가 좋은 캐시이다.

그러나 각 저장장치의 성능이 다른 이기종 저장장치 환경이라면 단순히 캐시 적중률만으로는 캐시의 성능을 평가할 수 없다. 만약, 전체 캐시 적중률은 높지만, 성능이 낮은 저장장치로의 접근에 대한 캐시 적중률이 매우 낮아, 해당 저장 장치에 대한 캐싱 효과를 보지 못하고, 직접 접근이 많이 일어난다면, 사용자가 느끼는 성능 하락은 심각해 질 수 있다.

그림 1은 일반적인 이기종 저장장치 환경을 나타내고 있다. 각 스토리지별 캐시 Miss Latency 가 다르기 때문에 이를 고려한 캐시 정책이 필요하게 된다.

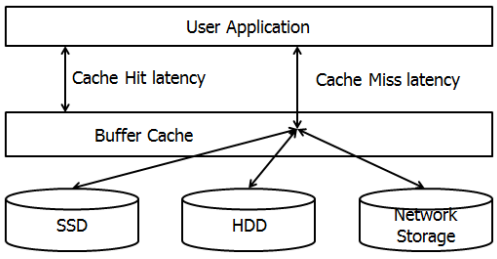

그림 1. 이기종 저장장치 환경
Figure 1. Heterogeneous Storage Environment

이와 비슷한 연구로 VDF[5] 가 있다. 이는 디스크 어레이 시스템에서 특정 디스크가 고장 난 경우 해당 디스크에 대한 접근 페널티를 다르게 하여, 디스크 복구 시간 및 접근 시간을 줄였으나, 해당 기법은 두 개의 운영모드(일반모드, 고장상태 모드) 변경 오버헤드가 있으며, 고장 모드에서만 제시한 알고리즘이 적용되는 한계점을 가지고 있다.

Ⅲ. 설 계

1. 이기종 저장장치 성능 척도

앞서 살펴본 것처럼 기존의 캐시정책은 하부 저장장치의 접근 시간이 동일하다는 가정 하에 만들어진 것으로, 이기종 저장장치의 특성을 반영하지 못한다. 이기종 환경에서는 캐시적중률이 높더라도 Throughput이 떨어지는 경우가 발생하기 때문에, 이를 높게 유지 시켜주는 새로운 캐시 관리 기법이 필요하다.

저장 장치의 성능은 크게 미디어 자체의 특성과 워크로드의 특성을 나눠지는데, SSD 와 HDD 의 예를 들면, 두 드라이브는 미디어적으로 입출력 성능이 다르며, 각 드라이브는 읽기, 쓰기, 순차접근, 임의접근등 워크로드 특성에 따라 다양한 성능이 나오게 된다. 본 연구에서는 이러한 다양한 성능 나타내는 척도로 Throughput을 사용하며, 이러한 특성을 제거 비용 그룹이라는 단위로 처리한다. 표 1은 SSD 와 HDD에서 4개의 그룹을 관리하는 경우를 나타낸 것이며, 각 그룹별 Throughput 이 다르기 때문에, 데이터가 캐시에서 제거될 때의 비용을 다르게 처리해야 효과적인 캐시운용이 가능한데, 이를 제거 비용으로 나타내고 있다.

표 1. 제거 비용 그룹 예제
Table 1. Example of eviction cost groups

그룹	특징	Throughput	제거비용
1	SSD, 순차읽기	160 MB/s	4
2	SSD, 임의읽기	120 MB/s	3
3	HDD, 순차읽기	60 MB/s	1.5
4	HDD, 임의읽기	40 MB/s	1

2. 캐시 관리 기법

표 1에서 나타난 것처럼 다양한 이기종 저장장치들은 미디어 특징과 워크로드 특징에 따라 산출되는 Throughput을 기준으로 캐시 제거 비용 그룹 단위로 나누어 질 수 있다. 본 연구에서는 이를 효과적으로 관리하기 위해서 복수개의 LRU queue와 하나의 제거 리스트를 사용한다.

그림 2는 제거 그룹이 4개가 있는 시스템에 대한 캐시 관리를 위한 LRU queue 구조이다.

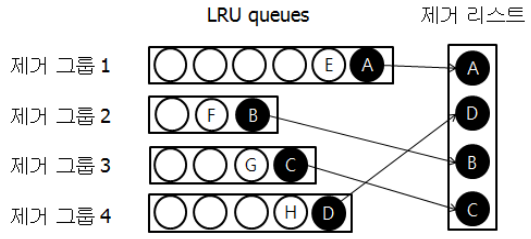

그림 2. 제거 비용 그룹 관리 자료 구조
Figure 2. Data structure for eviction cost group

각 제거 비용 그룹마다 독립적인 LRU queue를 유지하여, 해당 그룹에 접근이 일어나면, 순차적으로 LRU queue 에 유지가 된다. 각 엔트리는 queue 에 삽입될 때의 time stamp 값을 저장하며, 이는 추후 캐시 교체 작업에 사용된다. 각 그룹이 관리하고 있는 LRU queue 의 전체 크기는 시스템의 캐시크기 이며, 캐시 교체에 필요한 연산을 줄이기 위해 하나의 별도 제거 리스트를 유지 한다. 이 제거리스트는 각 LRU queue 의 마지막 엔트리들을 유지하고 있으며, 캐시 교체시, 제거 리스트에 있는 엔트리중 다음 비교 값이 가장 낮은 엔트리를 제거한다.

*비교 값 = 엔트리의 timestamp * 해당 엔트리가 속한 그룹의 제거비용*

각 LRU queue는 시간적 순서대로 엔트리가 존재하기 때문에 적어도, LRU queue 내에서는 접근 순서가 지켜지고 있다. 즉, 마지막 엔트리만 비교하는 것으로 캐시 교체를 하는 것이 결과적으로 전체를 비교하는 것과 동일한 효과를 가지게 되므로 교체 엔트리 검색 시간이 항상 “전체 제거 그룹의 개수“의 상수 값을 가지게 되므로 O(1)의 시간 복잡도를 가진다.

IV. 평 가

제안한 캐시관리기법의 성능을 평가하기 위해 그림 3과 같은 시뮬레이션 환경을 구축했다. 시뮬레이터의 워크로드 재생기는 지정된 워크로드와 동일한 I/O를 I/O 분배기로 계속 내려주며, I/O 분배기는 실험을 위해 SSD 와 HDD 에 대한 접근 비율을 지정된 파라미터에 따라 분배 해준다. 본 실험에서는 SSD : HDD 접근 비율을 0:100 / 20:80 / 40:60 / 60:40 / 80:20 / 100:0 과 같이 다양하게 셋업 하였다.

실험의 단순화를 위해 이기종 저장장치는 SSD 과 HDD 두 가지를 설정하였으며, 캐시 크기는 1024 MB 로 설정하였다. 또한, 제거 비용 그룹은 SSD와 HDD 두 개의 그룹만을 유지하며, SSD 의 throughput은 150 MB/s, HDD의 throughput은 50 MB/s 으로 각 그룹의 제거비용은 SSD = 1과 HDD = 3으로 설정하였다. 이는 SSD 의 throughput 이 3배 더 빠른 것을 의미한다.

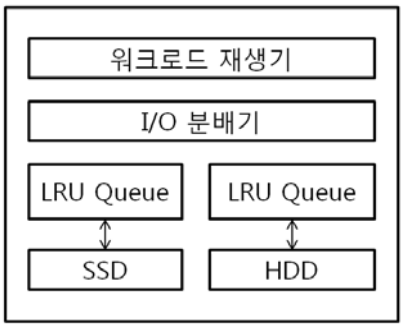

그림 3. 이기종 캐시 평가용 시뮬레이터
Figure 3. Heterogeneous cache simulator

본 논문에서 사용한 실험 워크로드는 MSR Cambridge 의 usr_0와 proj_0 [6] 워크로드를 사용하였다. 1024 Cache에서 usr_0 워크로드의 순수 LRU 적용시 hitratio 는 0.8 이며, proj_0의 순수 LRU 적용시 hitratio 는 0.45이다.

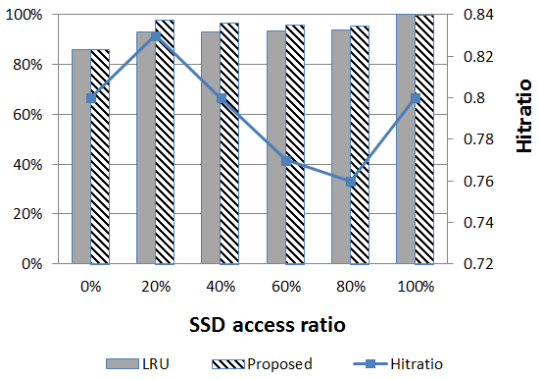

그림 4(a). MSR usr_0 workload
Figure 4(a). MSR usr_0 workload

그림 4(a)는 MSR usr_0 워크로드에 대한 결과

이며, (b)는 proj_0 에 대한 실험 결과이다. 왼쪽 세로축은 상대적인 throughput 이며, SSD에 100% 비율로 접근할 때의 throughput 이 100% 이다. 오른쪽 세로축은 Hitratio 이다. 제안하는 알고리즘을 적용하였을 경우 항상 LRU 보다 좋은 성능이 나오는 것을 알 수 있으며, 특히 그림 4(a)에서 SSD : HDD 접근비율이 20:80 일 때 30% 정도의 가장 큰 성능 향상을 나타내고 있다.

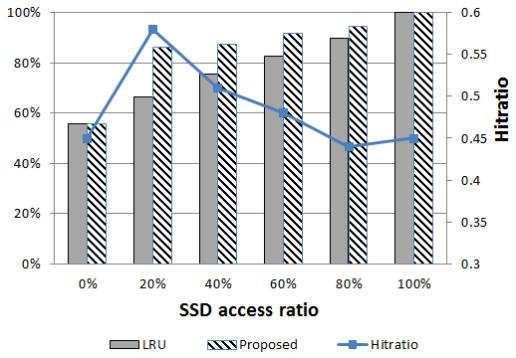

그림 4(b). MSR proj_0 workload
Figure 4(b). MSR proj_0 workload

그림 4(b)의 proj_0의 경우 SSD의 접근 비율구간이 20%에서 80% 까지 올라가면서 Hitratio 는 떨어지고 있지만, 오히려, throughput은 점점 더 증가하는 것을 볼 수 있다. 이는 이기종 저장장치에서 캐시를 유지하기 위해서는 기존의 캐시 성능 척도인 캐시 적중률만으로는 부족하다는 것을 보여주고 있다.

V. 결 론

본 연구는 기존의 캐시 관리 기법에서 고려하고 있지 않은 이기종 저장 장치에 대한 효과적인 캐시 관리 방법에 대한 내용을 설명하고 있다.

이기종 저장장치를 효과적으로 지원하기 위해 저장 장치들을 다양한 특성에 따라 제거 비용 그룹이라는 형태로 구별하며, 별도의 LRU queue로 유지하는 캐시 관리 기법을 제안하였다. 실험 결과 워크로드 특징에 따라 최대 30% 까지 성능향상이 있음을 보였다.

참 고 문 헌

[1] B. Pawlowski, S. Shepler, C. Beame, B. Callaghan, M. Eisler, D. Noveck, D. Robinson, and R. Thurlow. The NFS Version 4 Protocol. http://www.connectathon.org/talks97/index.html, 1997.

[2] IETF, Internet Small Computer Systems Interface (iSCSI), http://www.ietf.org/rfc/rfc3720.txt.

[3] CIFS, Common Internet File System, www.samba.org/cifs

[4] Amazon Simple Storage Service, http://aws.amazon.com/s3/

[5] Shenggang Wan, Qiang Cao, Jianzhong Huang, Siyi Li, Xin Li, Shenghui Zhan, Li Yu, Changsheng Xie, and Xubin He. 2011. Victim disk first: an asymmetric cache to boost the performance of disk arrays under faulty conditions. In Proceedings of the 2011 USENIX conference on USENIX annual technical conference (USENIXATC'11). USENIX Association, Berkeley, CA, USA, 13-13

[6] MSR Cambridge workloads www.snia.org