분산 오브젝트 저장 플랫폼의 스토리지 접근 워크로드 특성 분석

한유일*, 이은지*
*충북대학교 소프트웨어학과
http://oslab.cbnu.ac.kr

A Workload Analysis of Distributed Object Store to Backend Storage

Yuil Han*, Eunji Lee*
*Dept. of Computer Science, Chungbuk National University

요 약

디지털 데이터의 폭발적 증가와 형태의 다양화는 최근 비정형 대규모 저장 플랫폼의 급격한 확산을 이끌고 있다. 이러한 비정형 데이터 저장 시스템은 전통적인 파일시스템과 데이터를 저장 및 처리하는 방식이 상이하여 높은 성능을 위해서는 기존 하위 시스템의 최적화가 필요하다. 이에 본 논문에서는 최근 급부상하고 있는 분산 오브젝트 스토어인 Ceph 을 중심으로 오브젝트 스토어의 스토리지 접근 패턴을 분석하는 연구를 수행한다. 본 연구는 상위 계층의 접근패턴을 구체적으로 이해함으로써 차세대 데이터 플랫폼을 효율적으로 지원할 수 있는 스토리지 시스템을 개발하는 데에 기여한다고 하겠다.

1. 서론

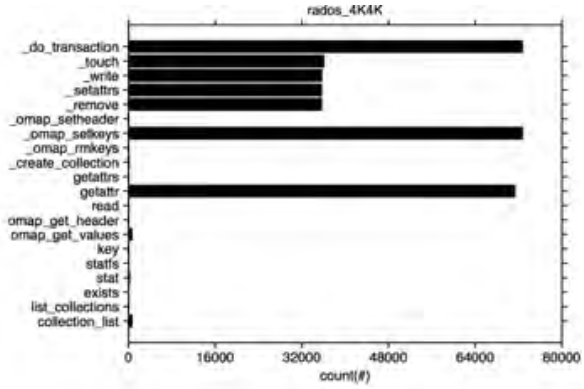
컴퓨팅 기술이 사회 전반에 깊숙이 활용됨에 따라 현대 사회는 과거 어느 때보다도 빠른 속도로 디지털 데이터를 생산해내는 빅데이터 시대를 맞이하고 있다. 이러한 데이터의 폭발적 증가는 대용량의 데이터를 효율적으로 저장하고 활용할 수 있도록 해주는 저장 플랫폼 기술의 급격한 성장을 이끌고 있다. 구글에서 개발한 GFS (Google File System), 아마존의 오브젝트 저장 시스템인 Dynamo, 대표적인 분산 데이터 처리 플랫폼인 하둡 (Hadoop) 등은 현재 다양한 분야에서 널리 사용되고 있는 데이터 처리 플랫폼이다. 한편 Ceph 은 최근 급부상하고 있는 분산 오브젝트 플랫폼으로 기존 시스템보다 확장성이 뛰어나고 다양한 인터페이스를 지원함에 따라 많은 인기를 얻고 있다[1].

이러한 분산 플랫폼들은 결국 실제 데이터를 하단에서 동작하는 로컬 스토리지에 저장하게 되는데 기존 사용자 어플리케이션과는 상이한 형태로 데이터를 저장하게 된다. 이에 본 논문에서는 분산 데이터 처리 플랫폼이 로컬 스토리지를 접근하는 워크로드를 분석하여 차세대 데이터 플랫폼을 효율적으로 지원할 수 있는 스토리지를 설계하는 데에 근간이 되는 자료를 제공하고자 한다.

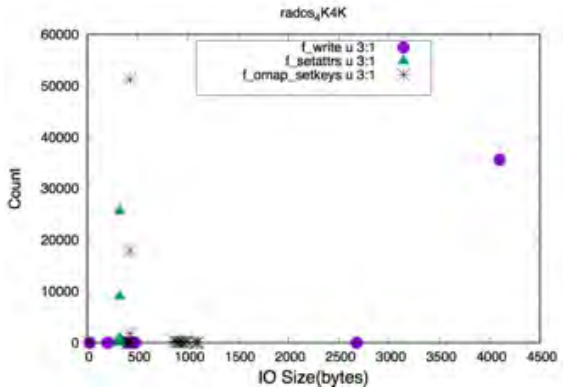
2. 오브젝트 스토어의 연산 분석

최근 데이터 플랫폼은 데이터의 형태와 양이 증가함에 따라 단순한 키밸류 형태로 데이터를 저장한다. 데이터를 구조화된 형태로 저장하는 전통적인 데이터베이스 시스템과 비교할 때, “오브젝트 형태의 저장” 방식은 확장성이 좋고 상이한 형태의 데이터를 동일한 인터페이스로 쉽게 저장할 수 있다는 장점이 있다. Ceph 은 데이터를 키와 함께 추가적인 속성을 저장할 수 있도록 해주는 오브젝트 스토어이다. 오브젝트 스토어는 사용자 계층에게는 PUT 과 GET 연산을 통해 데이터를 접근할 수 있도록 하지만, 내부적으로는 여러 개의 세부 연산을 통해 데이터를 처리한다. 본 논문에서는 Ceph 의 대표적인 벤치마크 프로그램인 RADOS 어플리케이션을 실행시키며 Ceph 에서 발생되는 세부 연산의 트레이스를 추출, 그 종류와 접근 패턴에 대해 분석하였다. RADOS 는 Ceph 의 다양한 서비스 하단에서 공통적으로 접근하는 Reliable Object Store 계층의 연산을 발생시키는 벤치마크로 타벤치마크의 동작을 포괄하는 벤치마크라고 할 수 있다. RADOS 를 통해 우리는 4KB 크기의 오브젝트를 60 초 동안 저장하도록 하였으며, 그림 1 은 해당 실행에서 발생한 세부 연산을 종류별로 보여주는 그래프이다.

그래프에서 보는 바와 같이 Ceph 은 모든 오브젝트 저장 연산을 트랜잭션 (_do_transaction) 으로 처리하는데 하나의 트랜잭션은 여러 세부 연산으로 구성된다. Ceph 의 오브젝트 저장 연산은 크게 세 가지 유형으로 구분할 수 있는데, 실제 데이터를 저장하는 write 연산, 데이터의 속성을 저장하는 _setattrs 연산, 그리고 오브젝트의 관리 및 검색을 위해 추가적인 데이터를 저장하는 _omap_setkeys 연산이 그것이다. RADOS 는 일정개수의 오브젝트를 생성하고 삭제하는 작업을 수행하는데, 그래프에서 보여주듯이 _write 와 _setattrs



(그림 1) 오브젝트 스토어의 세부 연산 분포

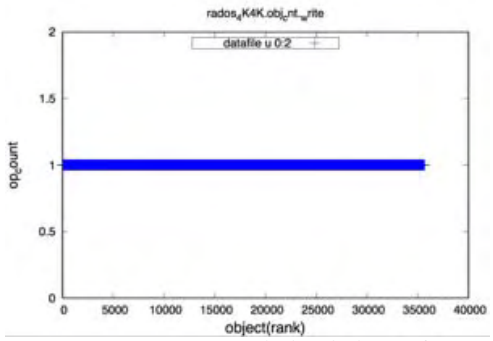


(그림 2) 연산 종류에 따른 I/O 크기 및 빈도수

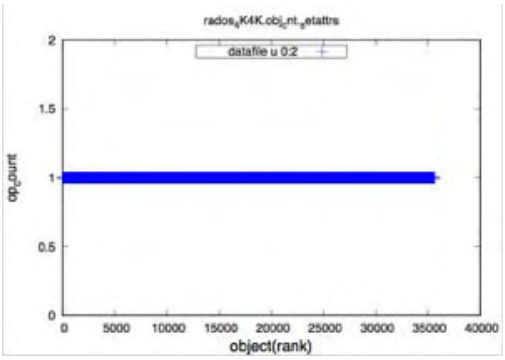
는 오브젝트 생성과 동일한 횟수로 발생하였다. 반면 _omap_setkeys 는 오브젝트 생성보다 훨씬 빈번한 형태로 발생한 것을 알 수 있다. _do_transaction 은 오브젝트의 생성와 삭제에서 모두 발생하기 때문에 _write 와 _remove 연산이 발생한 횟수를 합한 것 만큼 발생하였다.

그림 2 는 위의 연산 유형별로 쓰기의 크기 및 빈도수를 측정한 그래프이다. 그래프에서 두드러지는 결과는 오브젝트의 속성 (녹색 삼각형) 또는 검색용 키를 저장하는 연산 (별표) 은 대부분 500 바이트 이하이며, 대부분 1KB 를 넘지 않는 것으로 나타났다. 실제 오브젝트를 저장하는 write 연산은 4KB 가 가장 많은 것으로 나타났으나 이는 벤치마크를 4KB 크기의 오브젝트를 발생시키도록 설정한 결과이며 설정에 따라 그 결과는 바뀔 수 있다. 그러나 속성과 키를 저장하는 부분은 오브젝트 데이터의 크기와는 무관하게 작은 크기의 연산이 대부분인 것으로 관찰되었다.

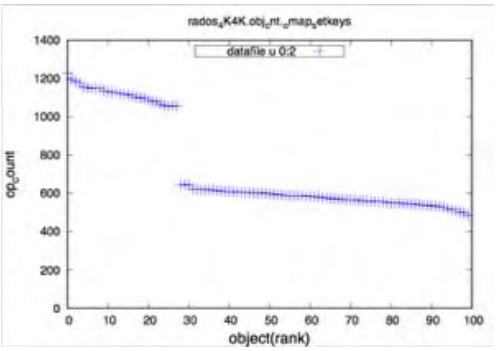
그림 3 은 연산 유형에 따른 접근의 지역성을 보여준다. x 축은 오브젝트를 접근횟수에 따라 정렬한 것이고, y 축은 해당 오브젝트에 대한 접근 횟수를 나타낸다. 그래프에서 보는 바와 같이 속성과 데이터에 대한 연산은 각 오브젝트에 각각 고르게 발생한 반면,



(a) 오브젝트 데이터 저장 연산



(b) 오브젝트 속성 저장 연산



(c) 오브젝트 키 저장 연산

(그림 3) 연산 유형에 따른 접근 지역성

키를 저장하는 연산의 경우에는 상당한 지역성을 나타내는 것으로 관찰되었다.

3. 결론

본 논문에서는 오브젝트 스토어에서 데이터를 저장하기 위해 발생시키는 세부 연산의 종류와 그 접근 특성에 대해 분석하였다. 각 연산 별로 상이한 빈도 및 접근 패턴을 나타내므로, 하위 스토리지 계층에서는 각 특성을 고려한 데이터 처리 시스템을 설계하는 것이 필요할 것으로 사료된다.

참고문헌

[1] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, C. Maltzahn “Ceph: a scalable, high-performance distributed file system,” Proceedings of the 7th symposium on Operating systems design and implementation (OSDI ’06).