토러스 구조를 고려한 객체의 인접 노드 분산 배치 및 이동 기법

박대준°, 신동군
성균관대학교 전자전기컴퓨터공학과
pdaejun@skku.edu, dongkun@skku.edu

Object Placement and Migration Method for Torus Network Based Distributed Storage System

Daejun Park°, Dongkun Shin
Department of Electrical and Computer Engineering, Sungkyunkwan University

요 약

분산 스토리지 시스템의 네트워크 구성중 하나인 토러스 연결망은 fat-tree에 비해 네트워크 구성에 적은 비용으로 구축이 가능하고, 확장성이 뛰어나다. 그러나 토러스 연결망 기반의 분산 스토리지 시스템에서는 노드의 구조적 위치에 따라 접근하기 위한 지연시간이 일정하지 않은 단점이 존재한다. 그러므로 토러스 연결망 기반의 분산 스토리지 시스템을 사용하는 분산 파일 시스템에서는 객체에 대한 접근 빈도에 따라 노드 배치가 이루어져야 지연시간으로 인한 문제를 피할 수 있다. 본 논문에서 제안하는 인접 노드 분산 배치 및 이동 기법은 토러스 연결망 기반의 분산 스토리지 시스템에서 자주 사용되는 객체를 LRU를 통해 구분하고, 자주 사용되지 않는 객체를 인접한 노드에 분산 이동시켜, 객체 간 hot/cold를 구분할 수 있다. 이로 인해 의사 난수 매핑에 비해 효율적인 노드 배치로 객체 접근에 대한 홉 수를 줄일 수 있다.

1. 서 론

빅 데이터 분석의 대중화와 대용량 멀티미디어 데이터 증가로 인해 대용량 스토리지에 용량에 대한 요구가 점차 커지고 있다. 각각의 노드가 스토리지를 제공하고, 이러한 노드가 서로 네트워크를 통해 연결되어 있는 분산 스토리지 시스템은 대용량 스토리지를 구성하기에 유리할 뿐만 아니라 새로운 노드의 연결을 통해 스토리지 용량의 확장이 용이하다는 장점이 있다. 분산 스토리지 시스템에서 각 노드를 연결하는 방법은 대표적으로 fat-tree와 토러스가 있다. Fat-tree 기반의 분산 스토리지 시스템은 노드의 개수가 늘어날수록, 네트워크 스위치 개수가 늘어나게 되어 비용 상 문제로 인해 확장성에 문제가 있다. 토러스[1] 연결망은 인접한 각 노드가 서로 연결된 구조를 가지고 있다. 그러므로 네트워크 스위치를 위한 추가적인 비용이 발생하지 않으므로, Fat-tree 기반의 분산 스토리지 시스템에 비해 확장성에 유리하다. 토러스 연결망에서 각 노드들은 주어진 축 상에서 자신의 이웃 노드와 연결되어 있고, 이에 추가로 반대편 모서리에 존재하는 노드와도 연결되어 있다. 토러스 연결망은 축의 개수에 따라 다양한 종류를 가지고 있는데, 2D 토러스 연결망은 기본적으로 메시 연결망과 유사한 구조를 가지고 있다. 토러스 연결망은 연결을 이루는 축마다 원형 고리 구조를 형성하고 있는데, 이로 인해 하나의 연결에 불량이 생기더라도 반대 방향을 통해 접근이 가능하다. 또한 각 축에서 가장 거리가 먼 두 노드가 연결되어 있으므로, 메시 구조에 비해 평균적으로 각 노드를 방문하기 위한 홉 수가 줄어들게 된다.

토러스 연결망은 노드가 서로 연결되어 있고, 특정 노드에 접근하기 위해서는 경로 상에 있는 모든 노드들을 방문해야 되기 때문에, 노드의 구조적인 위치에 따라 지연시간에 차이가 발생하게 된다. 그러므로 토러스 연결망을 사용하는 분산 스토리지 시스템은 노드 간의 지연시간 차이를 고려해야 한다. 분산 스토리지 시스템을 운영하는 대표적인 분산 파일시스템은 ceph[2]가 있다. ceph에서는 사용자에게서 발생하는 파일 입출력을 특정 단위인 객체로 구분하고, 이러한 객체들을 주어진 노드에게 배분하게 된다. 이 때 ceph에서는 객체들을 노드에게 분배하기 위한 알고리즘으로 CRUSH를 사용한다. CRUSH는 의사 난수를 사용하여 객체들을 노드들에게 균등하게 분배하고 복제함으로써, 특정 노드에서 장애가 생기더라도 복구할 수 있도록 하고 있다. 그리고 의사 난수를 사용하여 객체의 노드 매핑을 수행하기 때문에, 매핑 테이블을 유지하는 부하를 줄일 수 있다는 장점이 있다. 토러스 연결망을 가지고 있는 분산 스토리지 시스템에서 ceph를 사용하게 되면, CRUSH에 의해 객체가 노드들에게 분배된다. 이때 토러스 연결망 구조의 특징인 노드별 구조적 위치에 따른 지연시간 차이가 반영되지 않기 때문에, 특정 객체에 대해 예상되지 않은 지연이 발생할 수 있다. 그러므로 토러스 연결망을 가진 분산 스토리지 시스템을 위해, 노드의 구조적 위치에 따른 지연시간이 고려된 분산 파일시스템의 노드 분배가 필요하다. 본 논문에서는 3D 토러스 연결망을 가진 분산 스토리지 시스템에서, 토러스 연결망의 구조적 특성을 고려한 객체 배치와 이동을 위한 모델을 제안한다. 제안된 구조는 시뮬레이터를 통해, 기존 의사 난수를 통한 객체

이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2016R1A2B2008672)

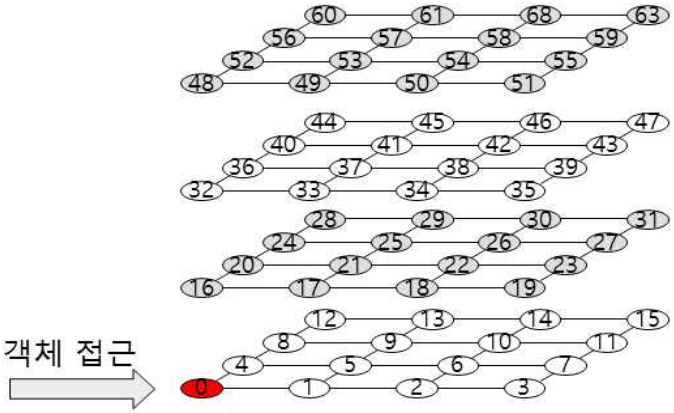

그림 1 토러스 연결망 기반의 분산 스토리지의 예

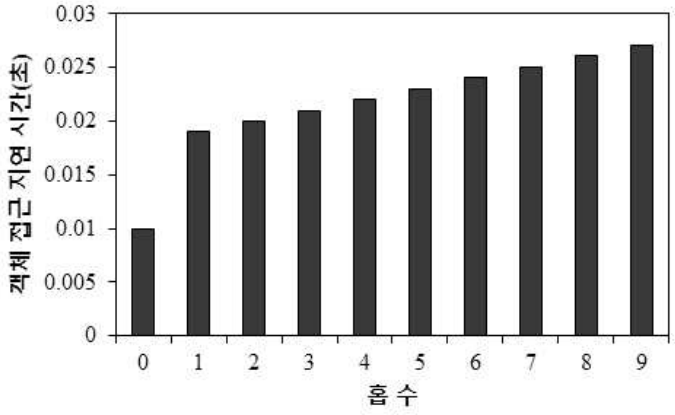

그림 2 토러스구조의 홉 수에 따른 객체접근 지연시간

배치와 비교하여 모델의 효과를 확인한다.

2. 관련 연구

분산 스토리지 시스템에서 접근 빈도에 따른 데이터의 hot과 cold를 구분하여 데이터에 대한 접근 시간을 줄이기 위한 대표적인 연구로, SSD를 캐시로 활용하는 기법들이 있다. S-RAC[3]는 방대한 양의 데이터를 관리하는 데이터 센터에서, 자주 접근되는 데이터를 구분하고 이에 대해 SSD로 캐시하여 스토리지 시스템의 성능을 향상시키기 위한 연구이다. S-RAC에서는 자주 접근되는 데이터를 구분하기 위해 데이터가 다시 접근되는 빈도인 reuse distance를 사용하고, ghost cache를 활용하여 특정 시간 내에 재접근되는 데이터에 대해서만 SSD에 캐시하는 기법을 사용하였다. 토러스 연결망의 특성 상, 자주 사용되는 데이터가 적은 홉 수안에 접근할 수 있는 노드에 존재하게 되면 지연시간이 줄어들게 된다. 그러나 이러한 노드에는 제한된 저장용량이 있으므로 자주 접근될 수 있는 데이터를 선별하여 저장해야 한다. 이러한 점에서 캐시의 대상이 되는 데이터를 선정하는 것과 유사한 요구 사항이 존재한다. 그러나 기존 연구에서는 토러스 연결망의 구조를 고려하지 않았으므로, cold로 구분된 데이터의 이동과 hot으로 구분된 데이터의 배치에 대해서는 적용할 수 없다.

3. 3D 토러스 연결망 기반의 분산 스토리지

그림 1은 본 논문에서 제안하는 토러스 연결망 기반의 분산 스토리지와 이를 사용하는 사용자 노드간의 연결 모델이다. 사용자 노드에서 발생하는 모든 입출력은 노드 0번으로 전달되고, 이 지점으로부터 사용자가 요청하는 객체의 위치에 해당하는 노드에 전달된다.

3D 토러스 연결망 구조에서 노드의 위치에 따른 지연시간 차이를 알아보기 위해, 홉 수를 늘려가면서 지연시간을 측정하였다. 측정에 사용된 토러스 연결망 시뮬레이터는 CODES[4] 시뮬레이션 엔진을 기반으로 개발하였다. 트레이스 기반으로 수행되는 토러스 연결망 시뮬레이터는, 객체에 대한 입출력을 입력으로 받고, 구성된 객체의 노드 배치에 따라 각 노드를 방문하면서 발생하는 홉 수와 지연시간을 시뮬레이션 한다.

실험에서 사용된 토러스 연결망은 6×6×6로 216개의 노드가 3D 형태로 구성되어 있다. 각 노드의 연결망은 1GB/s의 속도로 연결되어 있다. 또한 0번 노드에서 각 노드에게 보내는 객체의 크기는 4MB이다. 그림 2의 실험 결과를 보면 0번 노드에 대한 접근은 매우 짧은 시간 안에 이루어지지만, 홉 수가 늘어남에 따라 평균 지연 시간도 늘어나는 것을 볼 수 있다. 그러므로 분산 파일시스템에서 토러스 연결망을 효율적으로 사용하기 위해서는 자주 접근되는 객체에 대한 노드를 배치할 때, 홉 수가 적게 발생하는 위치에 배치해야 한다.

4. 토러스 구조를 고려한 인접 노드 분산 배치 및 이동

제안된 토러스 연결망 기반의 분산 스토리지를 효율적으로 사용하기 위해서는 사용자가 자주 접근하는 객체를 0번 노드에 토러스 구조에 가까이 두고, 자주 사용되지 않는 객체는 0번 노드와 멀리 두도록 해야 한다. 그런데 객체가 처음 접근될 때는, 객체가 자주 사용되는 정도를 알 수 없는 문제가 있다. 그러므로 객체가 처음 접근될 때는 우선 0번 노드에 저장한 뒤, LRU를 통해 자주 접근되지 않는 객체를 인접 노드로 이동시키도록 한다.

모든 객체가 0번 노드 또는 0번 노드와 인접한 노드에 저장되면 짧은 지연시간을 보장 할 수 있지만, 각 노드의 스토리지 용량은 한계가 있으므로, 각 노드에 저장되는 객체의 수가 제한된다. 그러므로 0번 노드에 객체들이 특정 개수 이상으로 저장될 경우, 이 객체들을 LRU를 통해 자주 접근되지 않았던 cold한 객체들을 인접 노드로 이동시킨다. 이 때 인접 노드 중에서 0번 노드와의 홉 수가 늘어나는 방향으로만 이동을 시킴으로써, cold한 객체가 저장되는 노드가 0번 노드와 멀어질 수 있도록 한다. 객체의 이동을 통해 cold 객체가 구분되게 되는데, 이동은 인접한 노드로 한정지어 이동으로 인한 부하가 적게 발생하도록 한다. 그리고 이러한 이동은 0번 노드뿐 만 아니라 모든 노드에서 발생한다. 토러스 연결망의 구조를 살펴보면, 0번 노드에서 인접한 노드들은 1 홉에 접근이 가능하다. 그리고 0번 노드에 인접한 노드에 다시 인접한 노드들의 경우, 0번 노드를 제외하고는 2 홉에 접근이 가능하다. 이런 식으로 자주 접근되지 않은 cold한 객체는 점점 LRU에 의해 이동 대상으로 선정되고, 이동을 통해 0번 노드에서 점점 홉 수가 늘어나는 노드에 저장된다.

그런데 working set이 변화함에 따라 자주 접근되지 않았던 cold한 객체의 접근 패턴이 변화하여 hot한 객체

로 바뀔 수 있다. 이러한 경우 hot으로 변한 객체 접근에 대한 지연시간이 늘어나게 되어 분산 스토리지 시스템의 성능을 하락시킬 수 있다.

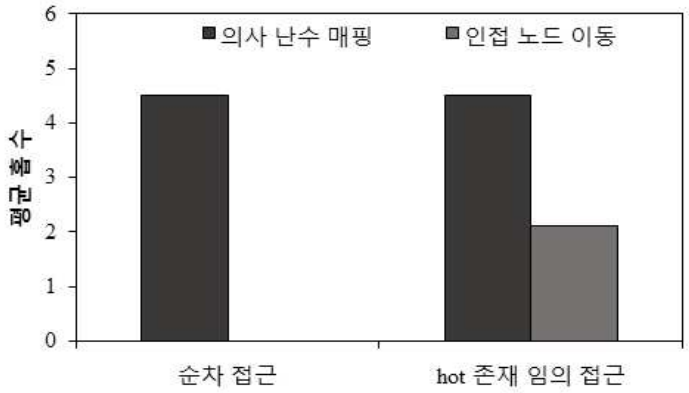

그림 3 워크로드에 따른 기법 별 객체접근 평균 홉 수

이를 해결하기 위해 자주 접근되는 객체에 대해 0번 노드와 근접한 노드로 이동시켜서 지연시간을 줄일 수 있도록 해야 한다. 각 노드에서 관리하는 LRU에서 2번 연속 같은 객체가 접근될 경우, 이 객체를 hot로 변한 객체로 판단하고 0번 노드로 객체를 이동 시켜, 자주 사용되는 객체의 접근에 대한 짧은 지연시간을 보장 할 수 있도록 하였다.

앞서 설명한 방법을 통해, 3D 토러스 연결망 기반의 분산 스토리지 시스템에서 자주 접근되는 객체는 0번 노드에 가까이 배치하고, 자주 접근되지 않는 객체는 이동을 통해 0번 노드와 점차 멀리 배치된다.

5. 실 험

제안된 배치 및 이동 기법을 검증하기 위해 앞에서 사용한 토러스 연결망 시뮬레이터를 사용하여 실험을 수행하였다. 토러스 구조는 6×6×6로 앞의 실험과 같은 형태이다. 각 노드의 연결망은 1GB/s의 속도로 연결되어 있고, 각 객체의 크기는 4MB이다. 실험에서 사용한 워크로드는 인위적으로 합성된 것으로 순차 접근, hot 객체가 존재하는 임의 접근이다. 트레이스 입력은 객체에 대한 접근 순서에 대한 것이다. 트레이스의 길이는 20000개의 접근으로 이루어져 있다. 제안된 배치 및 이동 기법과 비교하기 위한 객체 배치 모델은 ceph의 CRUSH와 유사하게 임의 난수를 사용하여 객체에 대한 노드의 위치를 결정하였다. 그림 3은 순차 접근과 hot 객체가 존재하는 임의 접근에 대한 각 기법의 평균 홉 수에 대한 비교이다. 순차 접근의 경우, 의사 난수를 사용한 모델은 4 홉이 넘는 평균 홉 수를 보이는 반면, 제안한 인접 노드 이동 기법의 경우 평균 홉 수가 0이다. 의사 난수의 경우, 순차 접근에 의해 모든 노드에 대한 접근이 이루어져 6×6×6 토러스 구조에서의 평균 홉 수가 나타났다. 제안된 인접 노드의 경우 매번 접근된 객체가 0번 노드에 배치되고, 이 후 접근되지 않는 객체는 다른 노드로 이동됨에 따라 항상 0번 노드에서 접근이 일어나게 된다. hot 객체가 존재하는 임의 접근의 경우, 의사 난수를 사용한 모델에서는 순차 접근과 비슷한 평균 홉 수를 보였고, 제안된 인접 노드 이동 기법에서는 약 2.1 홉 수를 보였다. 이는 인접 노드 이동 기법에서 접근 빈도에 따라 자동적으로 hot 객체를 0번 노드에 가까이 배치하였기 때문이다. 그림 4는 hot 객체가 존재하는 임의 접근 워크로드에서 접근 빈도 상위 10개의 객체에 대해 기법별로 접근 홉 수를 측정한 것이다. 의사 난수를 사용한 모델의 경우, 제안된 인접 노드 이동 기법에 비해 높은 평균 홉 수와 최대 홉 수를 보였다. 의사 난수를 사용한 모델의 경우 객체의 노드 배치가 고르게 이루어진 반면, 인접 노드 이동 기법의 경우 자주 접근되는 객체가 0번 노드와 가까운 노드에 배치되어 접근에 소요되는 홉 수가 줄어들게 되었다.

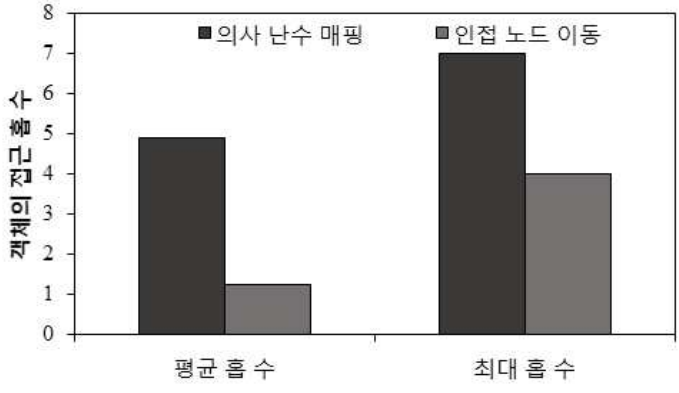

그림 4 접근빈도 상위 10개 객체의 접근 홉 수

6. 결 론

토러스 연결망은 fat-tree에 비해 적은 비용으로 구성이 가능하고, 확장성이 높은 네트워크 구성이다. 이를 활용한 분산 스토리지 시스템은 노드의 구조적 위치에 따른 지연시간의 편차를 극복 할 수 있도록, 저장되는 데이터의 접근 빈도를 통해 적절한 배치가 필요하다. 제안된 토러스 구조를 고려한 인접 노드 분산 배치 및 이동 기법은 이러한 문제를 해결하기 위해 우선적으로 적은 홉 수를 가지는 노드에 배치해 두었다가 LRU를 통해 보다 많은 홉 수를 가지는 노드로 이동시켜 접근 빈도에 따른 배치를 구현할 수 있도록 하였다.

참고문헌

[1] Blumrich, M., et al. Design and analysis of the BlueGene/L torus interconnection network. Vol. 3. IBM Research Report RC23025 (W0312-022), 2003.
[2] Weil, Sage A., et al. "Ceph: A scalable, high-performance distributed file system." Proceedings of the 7th symposium on Operating systems design and implementation. USENIX Association, 2006.
[3] Ni, Yuanjiang, et al. "S-RAC: SSD Friendly Caching for Data Center Workloads." Proceedings of the 9th ACM International on Systems and Storage Conference. ACM, 2016.
[4] Cope, Jason, et al. "Codes: Enabling co-design of multilayer exascale storage architectures." Proceedings of the Workshop on Emerging Supercomputing Technologies. Vol. 2011.