토러스 네트워크 기반 분산 스토리지 시스템의 데이터 배치에 따른 성능 분석

이하윤O, 신동군
성균관대학교 컴퓨터공학과
lhy920806@skku.edu, dongkun@skku.edu

Performance Analysis for Data Placement of Torus Network Based Distributed Storage System

Hayun LeeO, Dongkun Shin
Department of Computer Science and Engineering, Sungkyunkwan University

요 약

빅데이터, 클라우드, IoT 등의 발전은 대용량 스토리지 시스템(Storage System)에 대한 요구를 증가시키고 있다. 분산 스토리지 시스템은 다른 스토리지 시스템과 비교하여 낮은 비용과 높은 확장성을 갖추고 있기 때문에 대용량 스토리지 시스템으로써 적절하다. 최근 토러스(Torus) 네트워크 기반의 분산 스토리지 시스템이 제안되었다. 본 논문에서는 토러스 네트워크 기반의 분산 스토리지 시스템에 존재하는 특징을 식별하고, 이를 이용하여 3가지의 정책을 만들어 시뮬레이션을 통해 성능을 평가하였다. 이를 바탕으로 토러스 네트워크를 기반으로 하는 분산 스토리지 시스템을 위한 연구 방향을 제시한다.

1. 서 론

최근 빅데이터, 클라우드, IoT 등의 연구가 활발해지면서 이를 뒷받침하는 대용량 스토리지 시스템(Storage System)에 관한 관심이 커지고 있다. 이를 위해 상호연결망(Interconnection Network)을 이용한 분산 스토리지 시스템 연구가 진행되고 있다. 분산 스토리지 시스템은 NAS(Network Attached Storage)나 SAN(Storage Area Network)에 비해 낮은 비용과 높은 확장성으로 계속하여 증가하는 대량의 데이터 처리에 적합하다.

상호연결망은 네트워크 통신 및 고성능 컴퓨터를 구축하기 위해 많은 연구가 되어 왔다. 상호연결망 위상의 종류에는 링, 메쉬, fat-tree, 토러스 등이 있다. 특히 fat-tree와 토러스는 고성능 컴퓨팅에서 가장 많이 사용되고 있는 네트워크 위상이다.

이와 관련된 연구로 토러스 연결망을 이용한 분산 스토리지 시스템[1]이 제안되었다. 이는 토러스 네트워크가 fat-tree 네트워크와 비교하여 높은 확장성과 네트워크 스위치가 필요 없어 대용량 분산 스토리지 시스템 구축에 유리하기 때문이다. 하지만 노드와 노드 간의 네트워크 거리가 노드 간의 위치 별로 서로 다르며, 네트워크 홉이 늘어날수록 지연시간도 증가하는 단점이 있다.

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

본 논문에서는 토러스 네트워크를 기반으로 한 분산 스토리지 시스템에서 핫/콜드 데이터 배치 정책에 따른 성능의 변화를 시뮬레이션을 통하여 분석한다.

2. 토러스 네트워크

토러스 구조는 주로 병렬 컴퓨터 시스템을 구성할 때 여러 노드를 서로 연결하기 위해 사용하는 네트워크 위상 중 하나로 그림 1과 같이 메쉬 구조와 선형 구조의 특성을 합쳐 만든 구조이다. 토러스 구조는 주로 k-ary n-cube라고 표현하는데 이는 k^n개의 노드를 이용하여서 한 변이 k인 n차원의 도형으로 토러스 네트워크를 구성하는 것을 의미한다.

2.1 토러스 네트워크의 장점

먼저 메쉬 구조의 특성상 안전성이 높기 때문에, 이는 한 곳의 노드가 고장 나더라도 전체 시스템에 큰 영향을 미치지 않는다. 또한, 선형 구조와 같이 끝과 끝의 노드를 잇기 때문에 네트워크 홉 수를 절반으로 줄여 더 나은 성능을 제공한다. 이 외에도 fat-tree에 비해 저비용으로 높은 확장성을 가진다.

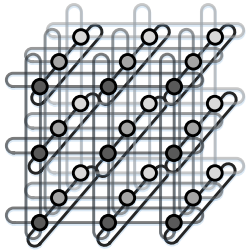

그림 1 3-ary 3-cube 토러스 구조

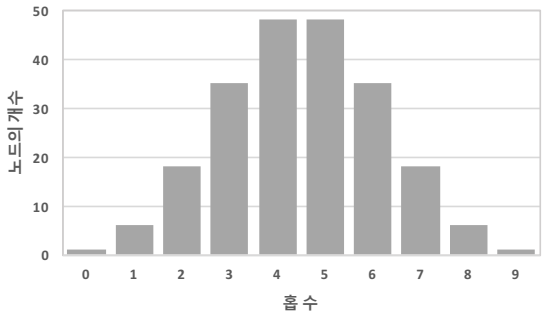

그림 2 6-ary 3-cube 토러스의 홉 수에 따른 노드의 개수

2.2 토러스 네트워크의 특징

토러스 네트워크는 fat-tree와 다르게 각 노드의 위치에 따라서 노드 간의 홉 수가 다르고, 홉 수에 따라 latency가 정비례하게 되는 특징이 있다. 그림 2는 6-ary 3-cube 토러스 네트워크에서 하나의 노드를 기준으로 홉 수에 따른 노드의 개수를 나타낸 그래프이다. 홉 수가 적거나 많은 노드는 매우 적고, 중간으로 갈수록 노드의 개수가 증가하는 것을 확인할 수 있다. 본 논문에서는 홉 수가 같은 노드들을 하나의 그룹으로 묶어 티어(Tier)라고 표현한다. 예를 들어, 홉 수가 4인 노드들의 그룹은 T_4가 된다.

3. 데이터 배치 정책

그림 3은 2-ary 3-cube 토러스를 사용하여 여러 노드에 데이터를 배치하는 정책을 보여준다. 그림 3 (가)는 워크로드에 포함된 데이터 객체를 핫/콜드 기준으로 정렬하고, 여러 노드를 티어에 따라 나눈 상태를 보여준다. 이때, 각 노드에는 스토리지가 있음을 가정한다.

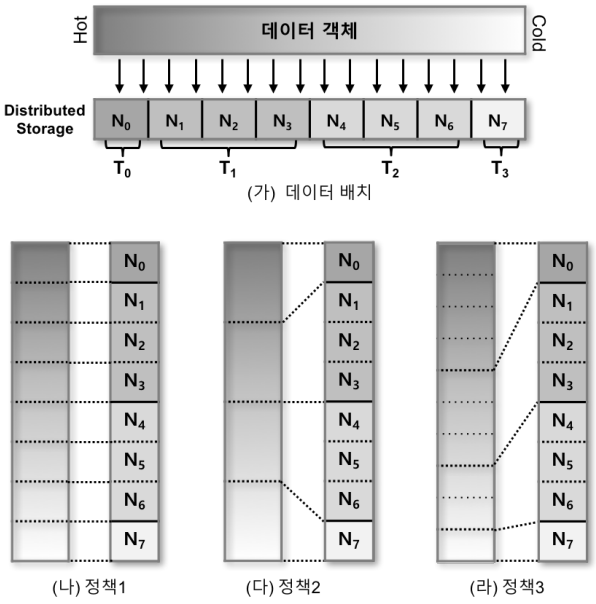

그림 3 데이터 배치를 위한 정책

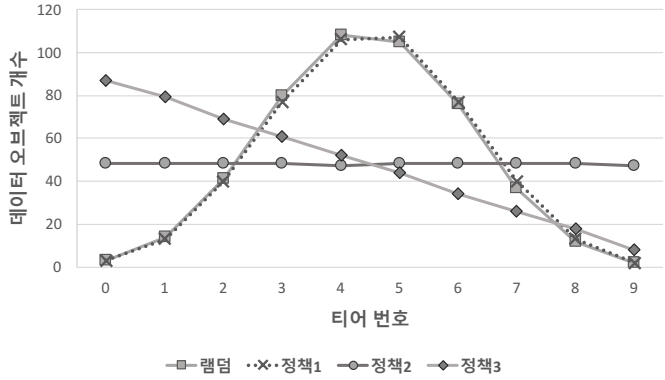

그림 4 티어에 배치되는 데이터 오브젝트 개수

그림 3의 (나), (다), (라)는 각 정책에 대한 데이터 배치 방법이다. 왼쪽과 오른쪽은 각각 데이터 객체 리스트와 노드들의 분산 스토리지를 나타낸다. 다음은 각 정책에 대한 설명이다. 정책1, 2, 3은 각각 노드의 수, 티어의 수, 그리고 티어가 낮을 수록 크게 부여한 가중치만큼 데이터 리스트를 분할하여 각 노드의 스토리지에 저장한다.

그림 4은 6-ary 3-cube 토러스 분산 스토리지 시스템에 워크로드를 이용하여 랜덤으로 배치했을 때와 위에서 설명한 정책들을 이용하여 배치했을 때 각 티어에 할당되는 오브젝트 개수를 나타낸다. 정책1과 랜덤은 배치되는 데이터의 분포가 비슷한 것을 확인할 수 있는데, 이는 랜덤으로 배치할 경우에 정책1과 같이 각 노드에 비슷한 수의 데이터를 배치하는 효과를 내기 때문이다. 정책2는 각 티어에 비슷한 수의 데이터를 배치하는데 이는 한 티어에 노드의 수가 적을수록 많은 데이터가 배치 됨을 의미한다. 정책3은 낮은 티어 번호에 더 많은 수의 데이터가 배치되어 T_9부터 T_4까지는 노드 당 배치되는 데이터의 수가 어느정도 균형을 유지하지만, T_4부터 T_0으로 갈수록 노드 당 배치되는 데이터의 수가 기하급수적으로 증가하여 결국 각 노드에 데이터가 불균형하게 저장되는 결과를 초래한다.

4. 실 험

4.1 시뮬레이터

본 논문에서는 토러스 네트워크 기반의 분산 스토리지에서 효율적인 데이터 배치를 시뮬레이션하기 위하여 ROSS[2]와 CODES[3]를 사용한다.

ROSS는 Rensselaer's optimistic simulation system의 약자로, 병렬로 이산사건(discrete-event)을 시뮬레이션 하는 타임 워프 시스템(Time Warp system)이다. 각 사건에 가상 시간(Virtual Time)을 부여하여 1개 이상의 프로세서에서 병렬적으로 수행한다. 하지만, 사건이 병렬적으로 수행되면 의존성 문제가 발생하여 시뮬레이션 결과가 달라질 수 있는데 이를 해결하기 위해 롤백을 사용한다. 문제가 발생한 것을 감지하였을 때, 미리 정의한 롤백 함수를 호출하여 사건을 실행하기 이전으로 되돌린

후 다시 시뮬레이션을 진행한다.

CODES는 ROSS 시뮬레이터를 이용하여 엑사스케일 스토리지 시스템의 성능 및 신뢰성을 시뮬레이션하기 위해 개발된 프레임워크이다. 토러스, dragonfly, fat-tree 등의 네트워크 모델이 구현되어 있으며, 시스템 디자이너가 네트워크를 구성하고 이벤트를 발생시킬 수 있도록 API를 제공한다.

시뮬레이션을 위하여 6-ary 3-cube 토러스 구조를 구성하고, 이 중 하나의 서버 노드에서만 클라이언트와 통신을 하는 상황을 가정하였다. 워크로드는 Hadoop의 terasort를 이용하여 생산하였고, 시뮬레이터에서 10μs 간격으로 메시지를 발생시켰다.

4.2 실험 결과

그림 5는 데이터 배치에 따른 메시지의 지연시간을 누적 분포 함수로 나타낸 그래프이다. 메시지의 지연시간을 측정할 때, 파일시스템 오버헤드를 제외한 네트워크 지연시간만을 고려하였다. 즉, T_0에 있는 노드에 접근하는 경우 지연시간은 0초가 된다. 그림 5의 그래프를 보면 랜덤 정책에서의 메시지 지연시간은 전체적으로 느린 반면, 정책3으로 갈수록 메시지의 지연시간이 감소하는 것을 확인할 수 있다. 이는 낮은 티어에 핫 데이터가 많이 배치되기 때문이다.

토러스 네트워크에서 하나의 메시지만을 보낸다고 가정할 때, 홉 수에 따라 지연시간이 비례하고, 같은 홉 수에서는 같은 지연시간을 갖게 되기 때문에 그림 5의 그래프에서처럼 누적 분포가 한 번에 많이 상승하는 것을 확인할 수 있다. 부드럽게 상승하는 부분은 지연시간이 긴 메시지들에 의해 대기 큐에 있는 다른 메시지들이 네트워크 지연시간 외에도 대기시간을 갖게 되어 다양한 지연시간이 나타나기 때문이다.

그림 6은 각 정책 별로 평균 홉 수와 평균 및 최대 지연시간을 나타낸 그래프이다. 랜덤에서 정책3으로 갈수록 평균 홉 수가 줄어들었다. 그 이유는 랜덤 정책에서는 무작위로 데이터를 배치한 것에 반하여, 정책1은 핫 데이터를 낮은 티어로, 콜드 데이터를 높은 티어로 균등하게 배치하여 자주 접근되는 데이터의 빠른 지연시간을 보장하기 때문이다. 정책2는 정책1처럼 데이터를 노드 별로 균등하게 배치하지 않고, 티어 별로 배치하였다. 이로 인해 낮은 티어에 데이터를 많이 배치하게 하여 평균 홉 수와 평균 지연시간은 감소하였지만 높은 티어에 배치되는 데이터도 늘어나게 되어 결과적으로는 최대 지연시간은 증가한다. 이를 보완하기 위해 정책3은 낮은 티어에 더 많은 핫 데이터를 배치하고, 높은 티어에 적은 콜드 데이터를 배치한다. 이는 노드 별 데이터 양의 불균형을 초래하지만, 낮은 평균 홉 수와 낮은 평균 지연시간을 보장한다.

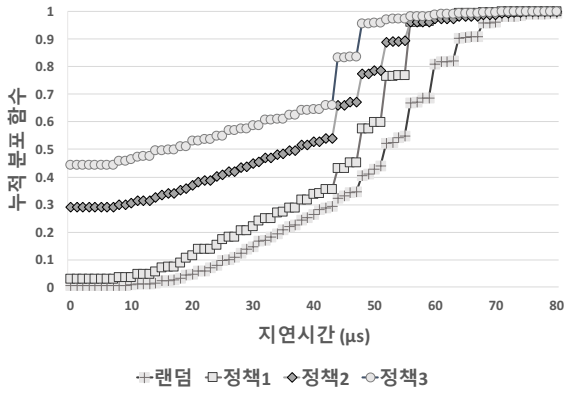

그림 5 정책에 따른 메시지 지연시간 누적 분포 함수

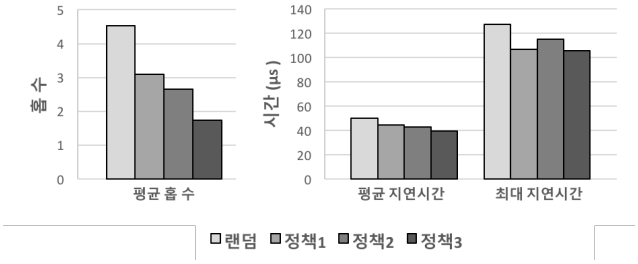

그림 6 정책 별 평균 홉 수 및 평균/최대 지연시간

5. 결 론

본 논문은 대용량 데이터 처리를 위한 시스템으로 토러스 네트워크 기반의 분산 스토리지 시스템을 시뮬레이션하였다. 데이터를 핫/콜드로 구분하여 랜덤으로 저장할 때와 3가지 정책을 사용할 때의 성능을 비교하였다. 홉 수가 낮은 티어에 많은 핫 데이터를 배치할수록 적은 지연시간을 보였다. 하지만 이는 각 노드에 배치되는 데이터의 불균형을 초래한다. 이를 해결하기 위해 낮은 티어에 가능한 많은 핫 데이터를 저장하고, 신뢰성을 위한 복제(replication) 데이터를 각 데이터의 다음 티어에 배치하는 정책을 취할 것을 제안한다.

참 고 문 헌

[1] C. Kim, D. Kim, H. Kim, Y. Kim, and D. Seo. “Torus Network Based Distributed Storage System for Massive Multimedia Contents”. Journal of Korea Multimedia Society, Volume 19, Issue 8, pp. 1487-1497, 2016.

[2] C. D. Carothers, D. Bauer, and S. Pearce. “ROSS: A high-performance, low-memory, modular Time Warp system”. Journal of Parallel and Distributed Computing, Volume 62, Issue 11, pp. 1648-1669. 2002.

[3] J. Cope, N. Liu, S. Lang, P. Carns, C. Carothers, and R. Ross. “Codes: Enabling co-design of multilayer exascale storage architectures”. In Proceedings of the Workshop on Emerging Supercomputing Technologies, 2011.