한국시뮬레이션학회 논문지
Vol. 16, No. 2, pp. 9-15 (2007. 6)

Group Master Cache를 활용한 SAN과 NAS의 통합 방안

이원복[1] · 박진원[2†]

Application of Group Master Cache for the Integrated Environment of SAN and NAS

Won-Bok Lee · Jin-Won Park

ABSTRACT

As the Internet grows and the mass multimedia data become popular, the storage system migrates from DAS, where the storage and the server are directly connected, to SAN and NAS. SAN connects the storages with a separate network, and NAS provides only file services, connects the storages with IP network. However, SAN and NAS can not fulfill the needs for companies if used separately, thus are asked to be integrated. In this research, we propose an efficient data sharing method which employees the concept of GMC, Group Master Cache for the integrated environment of SAN and NAS. GMC is based on MCI, Metadata server and Cluster system Integration, but tries to solve the high expansion cost problem with MCI. We introduce the basic concept of GMC, compare the performance of GMC with that of MCI using computer simulation.

Key words : SAN, NAS, Integration, Simulation

요 약

인터넷이 급성장하고 멀티미디어와 같은 대용량 데이터의 사용이 증가함에 따라 스토리지 시스템은 스토리지가 서버에 직접 연결되어 있던 DAS에서, 스토리지를 별개의 전용 네트워크로 연결한 SAN, 그리고 스토리지가 IP 네트워크에 연결되어 파일 서비스 기능만 전담하는 NAS로 발전하게 되었다. 그러나 SAN과 NAS를 독립적으로 사용하면 다양한 기업의 요구를 충족시키지 못한다. 그래서 등장한 것이 SAN과 NAS의 통합이다. 본 연구에서는 SAN과 NAS의 통합 방법의 하나인 MCI(Metadata server and Cluster Interation)를 바탕으로, MCI의 구현과 확장에 따른 비용 증가 문제를 해결하기 위해 GMC(Group Master Cahce)라는 개념을 제안하여 보다 효율적인 데이터 공유 방안을 제시한다. GMC의 기본 개념을 소개하고 기존의 MCI와 새로 제시하는 GMC를 컴퓨터 시뮬레이션을 통해 성능 측면에서 비교 분석하였다.

주요어 : SAN, NAS, 스토리지 통합, 시뮬레이션

1. 서 론

인터넷이 성장함에 따라 정보의 디지털화가 급속히 진행되고 있고 멀티미디어 데이터가 대량으로 신속하게 유통될 필요성이 증대되어 대용량 스토리지 시스템에 대한 요구가 그 어느 때 보다도 크게 확대되고 있다. 대용량 스토리지 시스템은 DAS(Direct Attached Storage)에서 SAN (Storage Area Network), NAS(Network Attached Storage)의 형태로 발전해 왔다[8,11]. 그러나 기존의 시스템들은 각각 단점을 가지고 있는데, SAN은 진정한 의미로 데이터 공유를 제공한다고 볼 수 없고, NAS는 파일형태의 전송과 LAN의 대역폭 부족으로 인해 다수의 사용자가 접근할 경우 병목현상이 발생할 수 있다[2]. 이와 같이 기존의 SAN과 NAS는 개별적으로는 고객의 다양한 요구를 모두 만족시킬 수 없다. 이러한 문제점들을 해결하기 위해 네트워크 스토리지 기술의 발전에 힘입어 기존의 SAN과 NAS를 통합하여[1] 하나의 대용량 스토리지 시스템을 구성하는 움직임이 활발히 진행되고 있다. 그 중 SAN 기반 데이터공유 방법의 하나인 MCI(Metadata server and Cluster system Integration)는 각 클라이언트들이 Cache

2007년 1월 12일 접수, 2007년 5월 22일 채택
1) 홍익대학교 전자전산공학과
2) 홍익대학교 게임학부
주 저 자 : 이원복
교신저자 : 박진원
E-mail; jinon@hongik.ac.kr

를 장착하여 시스템을 구축하였는데, 이 시스템은 클라이언트 설치 및 확장에 따른 비용증가라는 문제점이 있다.

본 연구에서는 기존의 데이터 공유방법인 MCI의 문제점을 해결하고자 GMC(Group Master Cache)를 활용한 데이터 공유 방법을 제안하고, MCI와 성능을 전송시간 측정 및 컴퓨터 시뮬레이션을 수행하여 비교 분석하였다.

본 연구의 구성은 2장에서 기존의 SAN과 NAS의 통합방법을 알아보고, 3장에서는 본 연구에서 제안하는 GMC를 활용한 데이터 공유방법을 설명하고, 4장에서는 컴퓨터 시뮬레이션을 수행하여 MCI와 GMC의 성능을 비교하며, 5장에서 결론을 요약하고 향후 연구 과제를 제시한다.

2. SAN 기반 데이터 공유

스토리지 시스템에서 안정적으로 자리잡은 기술은 SAN이다. SAN에서 스토리지들은 서버에 직접 연결되어 있지 않고 스토리지 전용 고속 네트워크로 연결되어 있다. 데이터 전송속도는 빠르나 서로 다른 OS를 사용할 경우 서로 다른 공간에 데이터를 저장하게 된다[6]. NAS는 스토리지가 LAN에 연결된 스토리지 시스템이다. 데이터 전송속도는 SAN에 비해 느리나 산업 표준 프로토콜인 NFS(Network File System), CIFS(Common Internet File System)를 사용해 서로 다른 OS라도 스토리지의 데이터를 공유할 수 있다. 이러한 SAN과 NAS의 장점을 활용하고, 단점을 보완하기 위해 SAN과 NAS를 통합하는 방법들이 제안되고 있다.

일반적으로 VOD(Video on Demand)와 같은 영상 데이터는 IP 네트워크를 통해 전송할 때 서비스 대역의 제약과 시스템의 CPU 부하 증대로 서비스 품질 저하를 유발시킬 수 있으며, 특히 멀티 시스템이 동일 IP 네트워크를 활용하게 되므로 서비스 확장, 고성능 서비스 품질 유지에 어려움이 발생할 수 있다. 이같은 문제점을 해결하기 위해 SAN 기반 데이터 공유 방법이[4] 제안되었는데, 메타데이터 관리 서버 구성 방법, 클러스터 시스템 방법 그리고 이 두 방법을 적절히 활용한 MCI가 있다[5].

2.1 메타데이터 서버를 이용한 데이터 공유

메타데이터는 데이터에 대한 내용, 구조, 특징 등을 기술한 정보로서 데이터에 대한 이해를 높이고 유용성을 판단할 수 있는 기준을 제공한다. 최근 웹의 발전으로 네트워크상에서 접근할 수 있는 데이터들이 증가하고 다양화되므로, 웹 데이터를 정확하고 신속하게 검색하기 위하여

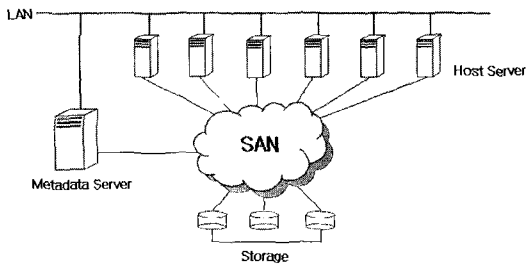


그림 1. 메타데이터 서버를 이용한 데이터 공유

이러한 메타데이터의 필요성이 증가하고 있다[3]. 이와 관련하여 메타데이터 관리 표준으로 널리 사용되는 Dublin Core[9] 외에, RDF(Resource description Framework)[13], INDECS(INteroperability of Data in E-Commerce Systems)[15], PRISM(Publishing Requirements for Industry Standard Metadata)[14] 등과 같은 XML 기반 메타데이터 관리 표준들이 제안되었다. 데이터에 대한 전반적인 정보인 메타데이터를 별도의 독립적인 메타데이터 서버에서 메타데이터를 운영 및 관리하게 함으로써 데이터에 대한 지능적인 관리를 가능하게 하고, 네트워크 내에 연결된 모든 이기종 서버에서 공통으로 데이터를 사용할 수 있도록 한다.

메타데이터 서버를 이용한 데이터 공유는 그림 1과 같다. 호스트 서버가 데이터 요청을 IP 네트워크를 통해 메타데이터 서버에 하면 메타데이터 서버는 작은 크기의 메타데이터를 회송하면서 스토리지 어느 곳에 목표 데이터가 위치하고 있는지 알린다. 그러면 이 메타데이터를 바탕으로 호스트 서버는 스토리지에 위치한 데이터를 SAN을 통해 직접 접근한다. 이러한 메타데이터 서버 방법은 NAS의 단점인 LAN의 병목 현상을 SAN을 이용하여 데이터에 직접 접근함으로써 해결할 수 있고, 또한 SAN의 단점인 데이터 공유를 메타데이터 서버를 이용함으로써 해결할 수 있다.

그러나 이러한 방법은 메타데이터 서버에 어떠한 문제가 발생할 경우 이 시스템에 연결된 모든 호스트 서버들이 스토리지에 접근할 수 없다는 문제점이 있다. 또한 모든 호스트 서버들이 스토리지에 있는 데이터에 접근하기 위해서는 메타데이터 서버에 있는 메타데이터를 가져와야 한다. 메타데이터 관리, 동시접근 제어 등은 메타데이터 서버에서 수행하기 때문에 이에 따른 추가 부담이 발생할 수 있다[4]. 메타데이터 서버 방식을 이용한 실제 구현 예로는 IBM Tivoli SANegy과 EMC의 Highroad이 있다[10,12].

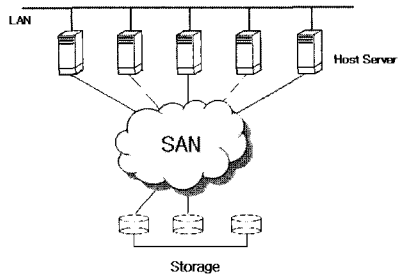


그림 2. 클러스터 시스템을 이용한 데이터 공유

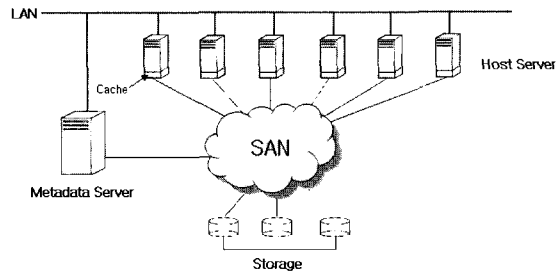


그림 3. MCI를 이용한 데이터 공유

2.2 클러스터 시스템을 이용한 데이터 공유

그림 2와 같이 클러스터 방식은 별도의 메타데이터 서버를 필요로 하지 않고, 각각의 호스트 서버들이 파일 시스템의 메타데이터를 유지하는 방식이다. 클러스터 방식은 어느 호스트 서버에 문제가 발생하더라도 다른 호스트 서버들에게는 아무런 영향을 주지 않는다. 이 방식은 호스트 서버들이 메타데이터 서버가 수행하는 메타데이터 관리, 동시접근 제어 등을 수행해야 한다. 데이터를 요청할 때 호스트 서버들은 자신들이 가지고 있는 메타데이터를 가지고 SAN을 통해 직접 스토리지에 접근해 작업한다.

이 방식에서 가장 먼저 고려해야할 점은 메타데이터 변화에 따른 데이터 일관성 유지 문제이다. 만약 하나의 호스트 서버가 스토리지에 있는 메타데이터를 수정할 경우 다른 호스트 서버들은 그러한 사실을 알지 못한다. 앞의 메타데이터 서버 방식은 메타데이터 서버에서 모든 데이터의 메타데이터를 관리하기 때문에 일관성 문제가 발생하지 않는데, 클러스터 방식은 이를 따로 관리하는 서버가 없기 때문에 일관성 문제가 발생한다. 이러한 메타데이터의 일관성 문제를 해결하기 위해서는 별도의 소프트웨어를 설치해야 한다. 클러스터 방식의 모든 호스트 서버들이 메타데이터 관리 및 일관성 유지 등의 문제를 고려해야하기 때문에 클러스터 방식을 실제로 구현하는 것은 다소 복잡해질 소지가 있다. 클러스터 방식을 이용한 실제 구현 예로는 VERITAS SANPoint가 있다[16].

2.3 MCI를 이용한 데이터 공유

그림 3과 같이 MCI는 앞의 메타데이터 서버 시스템을 기반으로 클러스터 시스템을 적절히 활용하여 구현한다[7]. MCI는 기존의 메타데이터 서버 역할은 그대로 두고 호스트 서버들이 로컬 Cache를 장착함으로써 메타데이터의 일부를 저장한다.

호스트 서버에서 데이터를 요청할 경우 우선 호스트 서버는 요청할 데이터의 메타데이터가 자신의 로컬 Cache에 있는지 검색하고 만약 존재한다면 그 메타데이터를 가지고 SAN을 통해 직접 데이터에 접근한다. 로컬 Cache에 요청하고자 하는 데이터의 메타데이터가 없다면 IP 네트워크를 통해 메타데이터 서버에 접속해 메타데이터를 가져온다. 여기서 호스트 서버들이 로컬 Cache에 저장한 메타데이터를 클러스터 방식처럼 계속 가지고 있는 것이 아니라, 호스트 서버들이 최신에 자주 사용한 데이터의 메타데이터만을 로컬 Cache에 저장한다. 메타데이터 서버는 메타데이터에 변화가 생길 경우 일관성 유지를 위해 각각의 호스트 서버들을 업데이트시킨다.

MCI는 메타데이터 서버 시스템과 클러스터 시스템의 단점을 어느 정도 해결해 준다. 우선 메타데이터 서버 시스템의 단점인 IP 네트워크 오버헤드는 각각의 호스트들이 로컬 Cache를 가지고 있기 때문에 보다 적은 횟수로 메타데이터 서버에 접근하게 된다. 또한 클러스터 시스템의 단점인 구현의 어려움은 메타데이터 서버가 호스트 서버에서 해야 할 역할 중 일부를 대신함으로써 해결된다. 메타데이터 서버와 호스트 서버간의 통신은 IP 네트워크를 통해 이루어지기 때문에 SAN 자원을 아낄 수 있다. 그러나 이 두 가지 방식을 구현하거나 확장하려면 추가 비용이 소요된다. MCI를 구현한 실제 예로 DirectNFS를 들 수 있다[7].

3. GMC를 이용한 데이터 공유

MCI는 메타데이터 서버만을 이용해 데이터를 공유하는 시스템보다 오버헤드가 적지만 일관성 유지와 동시접근 제어는 아직도 메타데이터 서버에 오버헤드를 발생시킨다. 또한 각 호스트 서버들이 로컬 Cache를 가지고 있기 때문에 로컬 Cache 장착 및 확장에 따른 비용이 증가한다. 이와 같이 MCI에서 발생하는 문제점을 해결하고자 본 연구에서는 GMC를 이용한 데이터 공유를 제안한다.

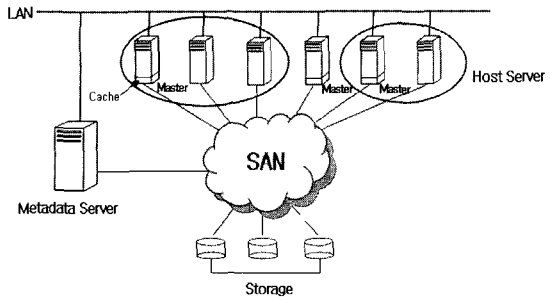


그림 4. GMC를 이용한 데이터 공유

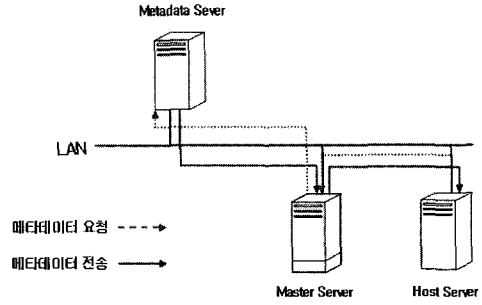


그림 6. 호스트 서버의 메타데이터 획득과정

메타데이터 서버의 일관성 유지에 따른 오버헤드와 로컬 Cache의 장착과 확장에 따른 비용 증가는 모든 호스트 서버에 로컬 Cache를 장착함으로써 발생한다. 이와 같은 문제는 그림 4에서 보는 바와 같이 몇몇 호스트 서버에만 로컬 Cache를 장착함으로써 해결할 수 있다. 이때 로컬 Cache를 장착한 호스트 서버를 Master 서버라 하며, 로컬 Cache를 장착한 Master 서버와 장착하지 않은 하나 이상의 호스트 서버를 묶어 Group이라 한다. 이렇게 Master 서버와 여러 호스트 서버로 구성된 시스템을 Group Master Cache(GMC) 시스템이라 하자.

GMC 시스템을 이용한 데이터 공유는 각 호스트 서버들과 메타데이터 서버 간에 IP 네트워크를 통해 서로 연결되고, 스토리지와 각각의 호스트 서버들은 SAN으로 연결된다. 메타데이터 서버는 SAN을 통해 스토리지에 있는 데이터들의 메타데이터를 통합 관리하고, IP 네트워크를 통해 Master 서버들의 로컬 Cache에 저장된 메타데이터들의 일관성을 유지한다.

3.1 Master 서버의 메타데이터 획득

GMC에서 Master 서버의 동작은 MCI 호스트 서버의 동작과 동일하다. 읽기 작업시 자신의 Cache를 검색하고 히트일 경우 바로 데이터를 스토리지로부터 가져오고, 미스일 경우 메타데이터 서버에서 메타데이터를 획득해 스토리지로부터 데이터를 가져온다. 쓰기 작업시에는 로컬 Cache를 검색하지 않고 바로 메타데이터 서버에 메타데이터를 요청한다. 단지 MCI와 다른 점은 Master 서버는 자신의 Group에 속한 호스트 서버들을 관리해야 한다는 점이다.

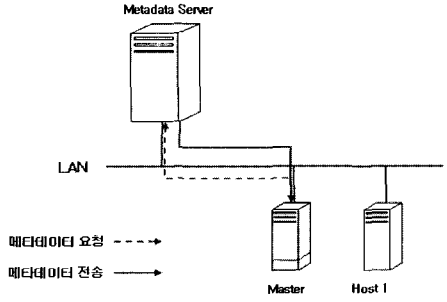


그림 5. Master 서버의 메타데이터 획득과정

3.2 호스트 서버의 메타데이터 획득

읽기 작업시 호스트 서버는 우선 Master 서버에 메타데이터를 요청한다. Master 서버 Cache에 메타데이터가 존재하면 그 메타데이터를 이용해 스토리지에서 데이터를 가져온다. 존재하지 않는다면 Master 서버는 호스트 서버에서 요청한 메타데이터를 메타데이터 서버에 요청해 자신의 Cache에 저장한 후 호스트 서버에 메타데이터를 전송한다. 쓰기 작업시 호스트 서버는 Master 서버에 메타데이터를 요청하는 것이 아니라, 직접 메타데이터 서버에 메타데이터를 요청한다. 이렇게 함으로써 데이터의 일관성 유지를 할 수 있다.

4. 실험 및 결과분석

SAN 기반 데이터 공유 방법 중 MCI와 본 연구에서 제안하는 GMC를 비교 분석하기 위해 Arena를 이용하여 MCI와 GMC 각각에 대한 컴퓨터 시뮬레이션 모델을 구축하고, 컴퓨터 시뮬레이션 모델을 실행하는데 필요한 입력 값을 얻기 위해 별도의 실험을 수행하였다.

4.1 실험환경

컴퓨터 시뮬레이션 모델의 입력값을 구하기 위한 실험에 필요한 장비는 메타데이터 서버, 호스트 서버, 이더넷(Ethernet) 네트워크이다. 메타데이터 서버의 사양은 Pentium

3 CPU 1.36GHz, RAM 1.0GB, 100Mbps 이더넷 카드이다. 호스트 서버의 사양은 Pentium 4, RAM 512MB, 100Mbps 이더넷 카드이다. 실험에 사용된 소프트웨어로는, 컴퓨터 시뮬레이션 모델링을 위해 Arena 9.0을, 전송시간을 측정하기 위한 네트워크 프로그램으로 Microsoft Visual Studio 2005를 각각 사용하였다.

4.2 시뮬레이션 모델링

SAN 기반 데이터 공유 방법 중 MCI와 본 논문에서 제안하는 GMC를 비교하기 위하여 Arena를 이용하여 MCI와 GMC 각각에 대한 컴퓨터 시뮬레이션 모델을 구축하고, 모델 실행에 필요한 입력 값을 얻기 위해 별도의 실험을 수행하였다.

MCI 시스템에 대한 시뮬레이션 모델은 메타데이터 서버의 읽기 요청시 Cache 미스일 때 메타데이터 전송, 쓰기 요청시 메타데이터 전송, 데이터 일관성 유지를 위한 메타데이터 전송 등 세가지 동작으로 이루어진다. 이 세가지 동작은 하나의 자원을 사용함으로써 하나의 메타데이터 서버가 동작하는 것과 같은 효과를 가진다. 데이터 일관성 유지에 필요한 데이터 값은 호스트 서버 개수에 따라 할당하였다.

GMC 시스템에 대한 시뮬레이션 모델은 호스트 서버와 Master 서버가 일정한 비율로 메타데이터를 요청하도록 구현하였다. 또한 Cache 일관성 유지를 위한 메타데이터 업데이트는 호스트 개수가 아니라 Master 서버 개수에 따라 데이터 값을 할당하였다.

4.3 실험방법 및 전송시간 측정

우선 호스트 서버가 메타데이터 서버에 메타데이터를 요청하여 전송받을 때까지의 평균시간을 측정하고, 메타데이터 서버에서 일관성 유지를 위해 각 호스트 서버들에게 수정된 메타데이터를 전송하는데 걸리는 평균시간을 실제 측정을 통해 구한다. 이렇게 구한 입력값을 이용하여 컴퓨터 시뮬레이션 모델을 동작시키고 이를 통해 읽기/쓰기 요청시 메타데이터를 획득하는 평균시간을 구한다. 또한 각 컴퓨터 시뮬레이션 모델이 작동하는 과정에서 로컬 Cache를 장착하고 있는 호스트 서버와 메타데이터 서버의 사용률을 구하여 비교 분석한다.

컴퓨터 시뮬레이션 수행에 필요한 입력 값을 측정하기 위해 메타데이터 서버 역할을 할 서버용 컴퓨터 한 대와 호스트 서버 역할을 할 PC 40대를 이용하여 네트워크 환경을 구축하여 전송시간을 측정하였다. 이때 전송 할 데이터의 크기는 4KB이다.

표 1. 데이터 평균 전송시간

호스트 서버의 갯수	평균 전송 시간 (μs)
1	1183
2	1271
3	1300
4	1505

표 1은 호스트 서버의 개수가 5대 미만일 경우만 나타낸 것이다.

그림 7은 호스트 서버가 5대부터 시작하여 5대씩 증가할 때마다 평균 데이터 전송시간을 측정하여 그림으로 나타낸 것이다. 또한 MATLAB 프로그램을 사용하여 표시된 점들로부터 선형방정식을 구했다.

$$y = 347.93*x + 44.607 \quad \text{(식 1)}$$

x : 호스트 서버의 갯수
y : 평균 데이터 전송시간

(식 1)을 이용해 호스트 서버 100대까지 평균 전송시간을 산출하여 시뮬레이션 모델에 적용시켰다.

4.4 시뮬레이션 결과 분석

MCI와 GMC의 성능을 비교하기 위해 호스트 서버에서 읽기와 쓰기 작업시 메타데이터를 획득하는데 걸리는 평균 전송시간을 구하고, 메타데이터 서버의 사용률과 로컬 Cache를 장착한 호스트 서버의 사용률을 구한다.

Cache 히트율을 50%, 60%, 70%, 80%, 90%로 하여 시뮬레이션을 수행하였으나 각각 비슷한 비율로 평균 전송시간이 증가하는 결과가 나왔다. 따라서 Cache 히트율은 70%로 고정하여 시뮬레이션을 수행하였다.

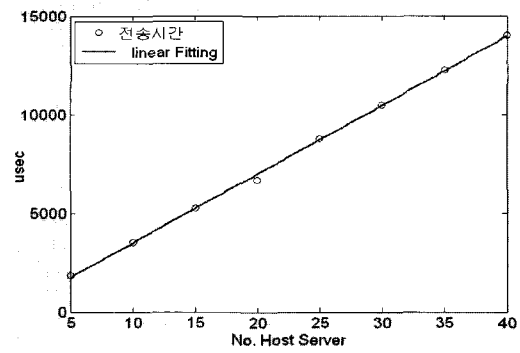


그림 7. 호스트 서버 개수에 따른 평균 데이터 전송시간

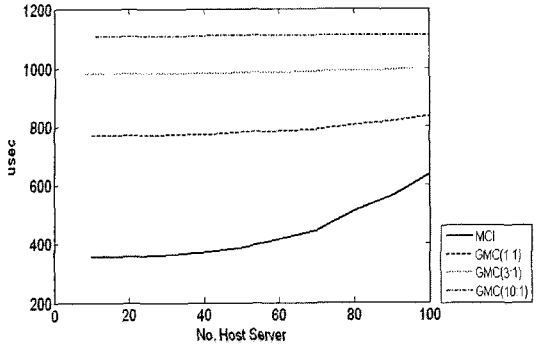


그림 8. 읽기 작업시 평균 메타데이터 획득시간

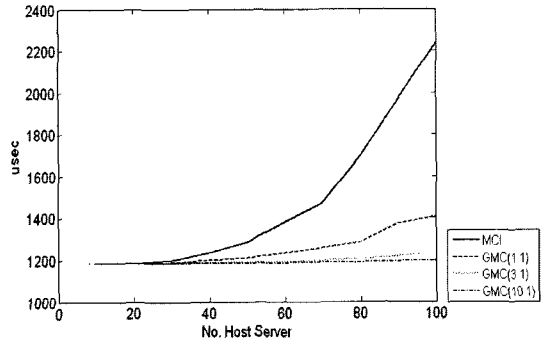


그림 9. 쓰기 작업시 평균 메타데이터 획득시간

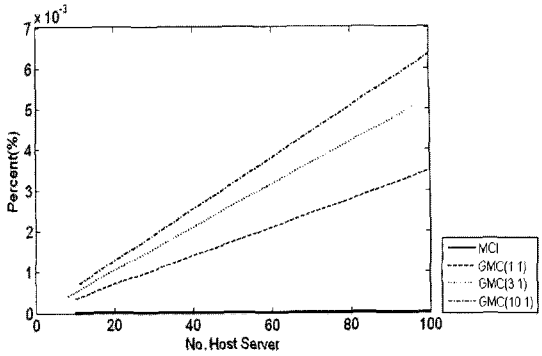


그림 10. 로컬 Cache를 장착하고 있는 서버 사용률

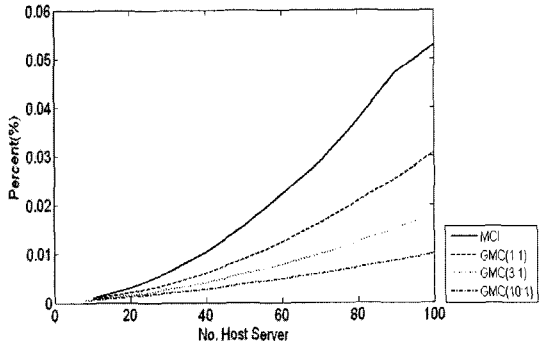


그림 11. 메타데이터 서버 사용률

그림 8과 같이 읽기 작업시 메타데이터를 획득하는 평균 시간은 MCI가 낮게 나타났다. GMC 호스트 서버는 읽기 작업시 메타데이터를 Master 서버에 요청하기 때문에 전송 시간이 MCI보다 높게 나타난다. 그러나 호스트 서버의 개수가 증가함에 따라 그 차이가 작아지고 있는데, 그 이유는 메타데이터 서버가 일관성 유지를 위해 할당하는 시간이 호스트 서버에 영향을 주기 때문이라고 판단된다.

그림 9에서 보는 바와 같이 MCI가 GMC보다 쓰기 작업시 평균 메타데이터 획득시간이 높게 나타났다. 쓰기 작업시 메타데이터 요청은 로컬 Cache를 이용하지 않고 메타데이터 서버에 요청하여 획득한다. 그렇기 때문에 메타데이터 서버가 일관성 유지를 위해 할당하는 시간이 길면 길수록 메타데이터 전송 지연이 발생한다.

그림 10은 메타데이터를 획득하는 과정에서 Cache가 장착되어 있는 호스트 서버들이 메타데이터 전송에 어느 정도 자원을 사용하고 있는지 보여준다. MCI의 경우 자신만이 로컬 Cache를 사용하기 때문에 호스트 서버의 자원 낭비가 거의 없다. 그러나 GMC의 경우 Group으로 묶여 있어 Master 서버에 다른 호스트 서버들이 접속하므로 Master 서버의 사용률이 증가한다. 증가 폭은 호스트 서버와 Master 서버의 비율이 커짐에 따라 커진다.

메타데이터 서버는 호스트 서버들이 읽기 작업시 Cache 미스가 발생하면 메타데이터를 전송하고, 쓰기 작업시 요청하는 메타데이터를 전송한다. 또한 쓰기 작업 완료시 수정된 메타데이터를 Cache가 장착되어 있는 호스트 서버에 업데이트시킨다. 그림 11은 메타데이터 서버의 자원이 MCI와 GMC에 따라 얼마나 사용되고 있는지 보여주고 있다. MCI가 GMC보다 사용률이 높게 나타난 이유는 전체 작업 중 쓰기 작업이 15%, Cache 히트율이 70%로 고정되어 있기 때문에 일관성 유지에 얼마나 많은 자원이 할당되느냐에 따라 메타데이터 사용률이 정해지기 때문이다.

5. 결론 및 향후 연구과제

본 연구에서는 SAN과 NAS의 통합 방법 중 하나인 MCI 방식에서 Cache 설치 및 확장에 따른 비용증가 문제를 해결하기 위해 새로운 시스템인 GMC를 제안하였다. 또한 성능상의 차이가 어느 정도인지 실험을 통해 GMC와 MCI를 비교 분석하였다. 실험 결과 GMC가 읽

기 작업시 평균 메타데이터 획득시간이 MCI 보다 높게 나타나지만 쓰기 작업시 평균 메타데이터 획득시간은 적게 나타났다. 또한, 로컬 Cache를 장착하고 있는 호스트 서버의 사용률은 MCI가 낮게 나타났지만, 메타데이터 서버의 사용률은 GMC가 낮게 나타났다. 그 이유는 메타데이터 서버가 데이터 일관성 유지를 위해 각 호스트 서버에 수정된 메타데이터를 업데이트시키는데 걸리는 시간의 차이가 주요 원인인 것으로 판단된다.

본 연구에서 제안한 Group Master Cache 데이터 공유 시스템은 MCI보다 Cache를 장착한 호스트 서버가 적기 때문에 Cache의 설치 및 확장에 따른 비용이 감소한다. 또한 메타데이터 서버 오버헤드의 주요 원인인 동시접근 제어와 일관성 유지에는 많은 도움이 될 것이라고 생각한다.

본 연구에서는 네트워크 전송시간만 고려하여 컴퓨터 시뮬레이션을 수행한 것이다. 향후 각각의 호스트 서버와 메타데이터 서버의 내부 처리시간이 전체적인 시스템에 어떠한 변수로 작용하는지 추가적인 연구가 필요하다.

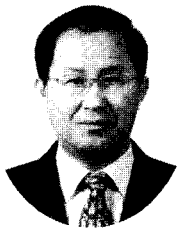
참 고 문 헌

1. 김기범 (2001), SAN과 NAS 통합에 관한 硏究, *단국대학교 정보통신대학원 석사학위논문*
2. 김동수 (2002), SAN(Storage Area Network), *세미나자료*, http://kdsoo.com/file/seminar/3.%20san.pdf.
3. 이미경 외 (2001), "웹사이트 관리를 위한 RDF 메타데이타 생성시스템", *정보과학회 논문지*, 제28권, 4호,
4. 성현희 (2006), "SAN+NAS의 다양한 통합 구성 방안", *On The Net*, 2006년 9월호
5. 황승희 (2005), "SAN과 NAS의 상호 보완적 통합", *On The Net*, 2005년 9월호
6. 황주영 (2003), "A SAN-based High Performance Shared Disk File System," *KAIST Ph.D. thesis.*
7. Bhide, A. et al, (2003), File Virtualization with DirectNFS, *HP 내부 자료*
8. Curtis, P.W. (2002) , "*Using SANs and NAS*," O'Reilly, Cambridge, U.S.A.
9. Dublin Core Metadata Initiative, http://dublincore.org
10. EMC (2002), Celerra HighRoad, http://www.emc.com
11. Gibson, G.A. and R.V. Meter, (2000), "Network Attached Storage Architecture," *Communications of the ACM*, Vol. 43, No. 11, pp. 37-45.
12. IBM, Tivoli SANegy (2002), http://www.ibm.com
13. Lassila, O. et al. (1999), "Resource Description Framework (RDF) model and syntax specification," *W3C*
14. PRISM WG (2001), "*PRISM: Publishing Requirements for Industry Standard Metadata*," PRISM Standard Org.
15. Rust, R. and M. Bide (2000), "The <indecs> Matadata Framework: Principles, Model and Data Dictionary," *EDItEUR*, WP1a-006-2.0.
16. VERITAS SANPoint (2001), http://www.symantec.com

이 원 복 (muble@nate.com)

2004년 홍익대학교 컴퓨터정보통신공학과 학사
2007년 홍익대학교 전자전산공학과 석사

관심분야 : 스토리지 시스템, 시뮬레이션, 컴퓨터 네트워크



박 진 원 (jinon@hongik.ac.kr)

1975년 서울대학교 공과대학 공학사 졸업
1982년 오하이오 주립대학교 산업시스템공학석사
1987년 오하이오 주립대학교 산업시스템공학박사
1987년~1988년 미국 남콜로라도 대학교 조교수
1988년~1999년 한국전자통신연구원, 책임연구원
2000년~현재 홍익대학교 계임학부 교수

관심분야 : 시스템 시뮬레이션, 컴퓨터 설계, 성능평가