# Request Redirection 기법을 통한 iSCSI 프로토콜 기반 스토리지 서버의 부하분산 방안 연구

서자원°, 최원일, Yuan Yang, 박영순
고려대학교 컴퓨터학과 인터넷 컴퓨팅 연구실
{jawon°, wonil22, yy, myongsp}@ilab.korea.ac.kr

# Request Redirection Method to reduce load for Storage Server based on iSCSI Protocol

Jawon Seo°, Wonil Choi, Yuan Yang, Myongsoon Park
Internet Computing Laboratory, Information and Communication Department, Korea University

요 약

유비쿼터스 환경에 대한 관심과 휴대용 컴퓨팅 기기의 보급으로 가까운 미래의 사람들은 언제 어디서나 네트워크를 통해 데이터 접근이 가능하게 될 것이다. 특히 영화나 음악과 같은 멀티미디어 데이터의 폭발적인 증가로 인해 확장성 있는 네트워크 스토리지 시스템의 필요성이 부각되고 있다.

SAN(Storage Area Network)은 높은 확장성과 빠른 속도를 제공하여 엔터프라이즈 환경에 적합한 스토리지 네트워크 시스템이다. 최근에 SAN 환경은 SCSI Architecture Model(SAM)의 표준으로 채택된 iSCSI를 이용한 IP기반의 SAN으로 옮겨가고 있다.

본 논문에서는 iSCSI 기반의 IP SAN환경에서 서비스 클라이언트가 증가함에 따라 스토리지 서버의 부하가 커지는 문제를 해결하기 위해 스토리지 디바이스에서 클라이언트로 데이터를 직접 전송하는 방안을 제안한다.

## 1. 서 론

디지털 라이프 패러다임은 이미 우리의 삶에 많은 영향을 미치기 시작했다. 우리는 하루도 빠짐없이 컴퓨팅 기기를 사용하고 있으며, 그 단적인 예는 휴대폰을 통해 영화를 보고 음악을 들을 뿐만 아니라 언제 어디서나 인터넷에 접근 할 수 있다. 이러한 디지털 라이프 패러다임은 멀티미디어 지향적인 데이터의 폭발적 증가를 야기하였고 기업은 확장성 있는 스토리지 시스템을 지원할 수 있는 효율적인 방법을 필요로 하게 되었다.

SAN은 스토리지 서버를 스토리지 디바이스에 연결시키고 데이터를 전송하는 네트워크다. SAN은 확장성이 뛰어나서 스토리지와 네트워크를 증설함으로써 대용량 스토리지에 대한 요구를 충족시켜 왔다[1].

iSCSI는 데이터 스토리지 장비들을 묶어주기 위해 스토리지 네트워크 표준에 기반하여 Internet Engineering Task Force(IETF)에 의해 개발된 인터넷 프로토콜이다. iSCSI는 SCSI 커맨드를 캡슐화하여 TCP/IP 네트워크 프로토콜을 이용해 전송한다. iSCSI를 이용해 IP기반의 SAN 구성이 가능하다[2]. iSCSI를 이용해 IP 기반의 SAN을 구성하게 되면, 스토리지 자원이 IP 네트워크 상에 위치할 수 있으며 스토리지 자원의 가용성과 확장성을 높일 수 있다.

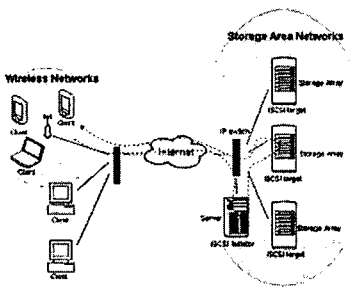


그림 1. 클라이언트와 SAN간의 통신

그러나 스토리지에 접근하기 위해서는 파일 시스템을 거쳐야하고 이러한 파일 시스템은 서버에 위치해 있기 때문에 추가적으로 스토리지를 설치하기 위해선 서버의 능력을 고려해야 한다. 즉, 클라이언트와 스토리지 디바이스 사이에서 발생하는 모든 커뮤니케이션은 서버를 통해 이루어진다. 이러한 구조에서는 서버가 병목 지점이 되어 SAN의 확장성을 제한하게 된다. 이 문제를 해결하기 위해서는 더 많은 서버를 구입해야 하지만, 근본적인 해결책이 되지 못한다. 따라서 IP-based SAN의 확장성을 지원하기 위하여 서버의 병목현상을 해결할 수 있는 새로운 방안을 제안한다.

## 2. 연구 배경

iSCSI는 TCP상에 올라가는 SCSI Command의 베어러 역할을 수행하는 프로토콜이다. 또한 표준 SCSI 아키텍처 모델과의 완전한 호환성을 제공한다. SCSI는 스토리지 디바이스와의 I/O를 수행하기 위해 사용되는 업계 표준 프로토콜이다. 스토리지 시스템이 버스 구조에서 네트워크 구조로 옮겨감에 따라 Fiber Channel SAN과 같이 SCSI를 네트워크 전송 프로토콜에 기반하여 사용하고 있다[1]. 또한 최근IP 네트워크의 성능이 높아짐에 따라 IP 기반의 SCSI전송 프로토콜에 대한 연구가 진행되고 있다[3][4].

## 3. Direct Data Transmission (DDT)

IETF에 의해 2000년 2월에 iSCSI가 제안된 이후 iSCSI는 공식적인 스토리지 표준 프로토콜이 되었고 계속해서 발전하고 있다. 이후 성능향상을 위하여 "Phased-Collapse" 필드와 "Immediate Delivery" 필드가 iSCSI Protocol에 추가되었다[5].

본 논문에서는 성능 개선을 위해 Reserved Field를 이용한 DDT(Direct Data Transmission) 기법을 제안한다. 제안 방안을 위한 가정 사항은 다음과 같다.

- 클라이언트는 TCP/IP 네트워크를 통해 스토리지 디바이스에 직접 접근할 수 있다.
- 스토리지 디바이스는 FTP, HTTP와 같은 일반적인 데이터 전송 프로토콜을 지원한다.
- 제안 방안은 Read I/O 요청에만 적용한다.

Write I/O 요청의 경우 R2T(Ready To Transfer), Error Recovery와 같은 iSCSI 고유의 메커니즘을 사용해 안정적으로 데이터를 전송할 수 있지만 Read I/O 요청은 리얼타임의 특성을 가지기 때문에 서버는 제한된 시간 안에 사용자에게 서비스를 제공해야 한다. 본 논문에서 제안하는 DDT 기법은 iSCSI Read 커맨드의 응답으로 Storage Device에서 클라이언트로 요청한 데이터를 보낼 때 서버를 거치지 않고 직접 전송한다.

### 3.1 Text Parameters

iSCSI 프로토콜에서 다양한 메커니즘에 대한 파라미터 값의 설정을 위해 Text Request, Response PDU(Protocol Data Unit)를 이용한다.

iSCSI 프로토콜은 약속된 기능의 사용과 파라미터 값 설정을 위하여 Text Request를 이용해 해당 값들을 전송하고 상대측에서 Text Response PDU를 통해 응답을 한다. DDT를 지원하기 위한 파라미터 설정 또한 iSCSI Text PDU를 이용한다. 이는 iSCSI Protocol에 이미 정의된 사항이므로 소스코드의 수정이 아닌 단순히 새로운 파라미터 값만을 추가함으로써 제안하는 성능개선 방안을 구현할 수 있다.

그림 2는 DDT을 지원하기 위해 새롭게 추가된 파라미터 값을 나타낸다. DDT 사용여부를 Yes로 설정하고, 클라이언트와 서버에서 사용된 Application의 종류, 클라이언트를 직접 접근하기 위한 IP 주소와 클라이언트에 의해 요청된 파일의 크기와 생성된 iSCSI Command들을 식별하기 위한 파일 tag 값들을 보여 준다.

블록 I/O 기반 의 iSCSI 프로토콜 메커니즘에선 한 파일의 I/O를 위해 많은 iSCSI Command가 생성될 것이고 Command들에 의해 생성된 블록들이 동일한 파일에 대한 것인지를 식별하기 위해 유일한 File Tag 인자를 사용한다.

DirectDataTransmission = Yes
ApplicationProtocol="FTP"
ClientIP_Address="123.123.123.1"
ClientTCP_Number="5003"
FileSize = 16375000 bytes
Initiator File Tag = 0xabc4

그림 2 Text Parameters

### 3.2 iSCSI SCSI Command PDU

DDT를 지원하기 위해서는 기존 iSCSI Command PDU의 새로운 기능 추가가 필요하다. DDT에서 사용될 정보는 기능 추가를 위해 남겨진 예약된 필드를 이용하기 때문에 기존 iSCSI 프로토콜과 호환성을 갖는다.

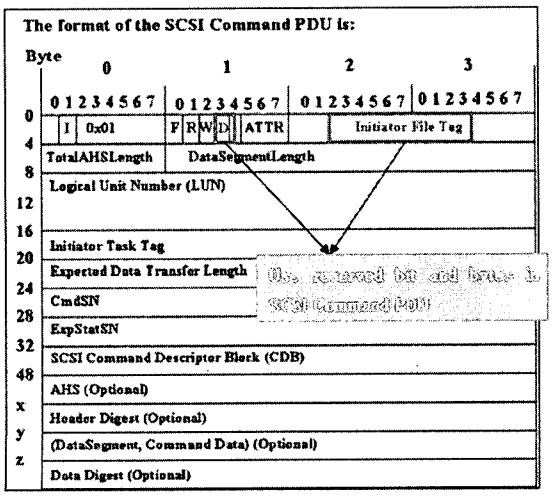


그림 3 iSCSI SCSI Command PDU Format

그림 3은 DDT 구현을 위해 사용하는 필드를 나타낸다. 각 필드의 세부 기능은 다음과 같다.

① D (Direct Data Transmission) bit : 1 로 세팅 되었을 경우 스토리지 장치의 target은 요청된 블록을 서버의 Initiator에 전송 하지 않고 자신의 Local File Buffer에 저장한다. Local File Buffer는 DDT의 text request에 의해 생성된 파일 크기의 버퍼이다.

② Initiator File Tag : Text request에 의해 보내진 파일에 대한 유일한 식별자이다.

③ AHS(Additional Header Segment) Type : 추가된 새로운 기능을 위해 0x03 값을 사용한다. 즉 해당 필드의 값이 0x03일 경우 다음 byte는 File Fixed Offset을 가리킨다.

④ File Fixed Offset : Local File Buffer 내의 해당 블록의 offset 위치를 가리킨다.

### 3.3 Direct Data Transmission의 동작과정

Local Disk를 사용하지 않는 네트워크 스토리지 기반 서버는 Client의 파일에 대한 I/O를 수행하기 위해 네트워크 스토리지에 접근해야한다. 서버에 접속하는 Client의 수가 증가하고 서버와 네트워크 스토리지 간에 스토리지 트래픽이 증가하게 되면 서버는 병목 지점으로 작용하며 전체 시스템의 성능은 급격하게 떨어진다. 그림 6은 DDT를 지원하는 서버가 Client의 데이터 Read에 대한 QoS를 만족시키지 못할 경우 iSCSI Text Request PDU를 통해 네트워크 스토리지가 DDT방식으로 동작하는 과정을 보여준다. Text Request는 스토리지가 직접 Client에 접근하기 위한 정보(Application Type, Client 주소, 파일 크기, File Tag 등)들을 전송 한다. 스토리지의 target은 응답 메시지로 iSCSI Text Response를 전송하고 동시에 Client와 직접 Connection을 맺는다. 스토리지의 target은 서버의 Initiator로부터 D flag가 체크된 iSCSI SCSI Command read PDU를 수신 하면 File Tag 와 File Fixed Offset 정보를 통해 해당 블록들을 File Buffer에 수집한다. 해당 파일에 대한 블록들이 모두 모였을 경우 Target은 직접 Client에게 해당 파일을 전송한다. 그 후 서버가 Initiator에게 iSCSI SCSI Response를 전송함으로서 Task를 마치게 된다.

DDT 방식을 적용하는 경우 Text Request, Response 등의 추가적인 PDU를 생성하고 전송하는 overhead가 발생하지만, 이는 Data I/O에 비해 PDU의 크기가 상당히 작다.(iSCSI Protocol 헤더 48bytes * 2, Client 정보가 포함된 Text) 이를 통해 서버의 Bottle Neck 문제를 효과적으로 회피할 수 있다.

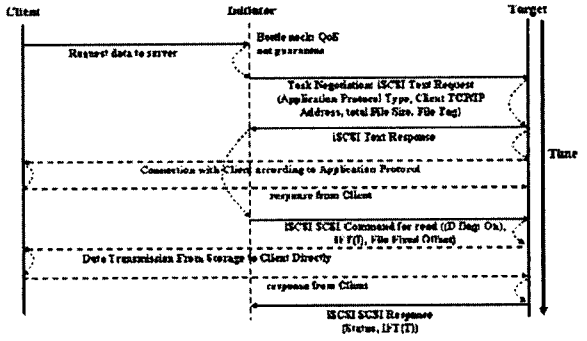


그림 4. DDT 동작방식 Sequence Diagram

## 4. 성능 평가

본 장에서는 본 논문에서 제시한 DDT 방안을 적용한 시스템을 구현하여 성능을 측정하고 그 결과를 비교 분석하였다. 성능 측정을 위해 설정한 실험 환경은 다음 표 1과 같다.

표 1. 실험 환경

| | Storage Server | Client |
|---|---|---|
| Device | Linux Server | PDA(LG AIV+) |
| CPU | P IV 1.5 GHz | Intel Strong ARM SA-1110 |
| Memory | 256MB | 32MB |
| OS | Linux 2.4.18 | Windows CE |
| Network Interface | 100Mbps | CDMA-2000 1X(14.4Kbps) |

실험 방법은 I/O Request Data Size를 512Byte ~ 64KByte로 증가 시키며 동시 서비스 클라이언트 수 1,4,8,16개에 대하여 각각 평균 Latency를 측정하였다. 그림 5~8을 통해 일반적으로 Data Size가 증가함에 따라 Latency가 증가하는 것을 볼 수 있다.

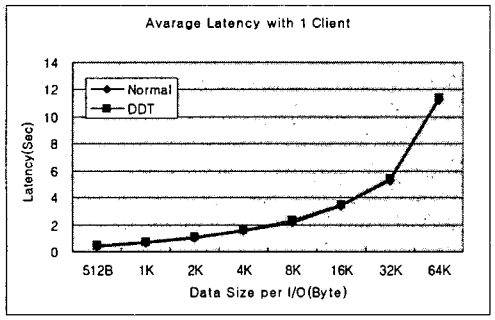


그림 5. Average Latency with 1 Client

클라이언트 1개의 경우 서버 부하가 거의 없는 상태에서 DDT적용 여부에 상관없이 비슷한 Latency값 을 나타냈다. DDT를 적용한 경우 오히려 약 0.1초 정도 Latency가 증가 하였는데 그 원인은 DDT방안에 의해 추가된 메시지의 왕복 Latency 때문이다.

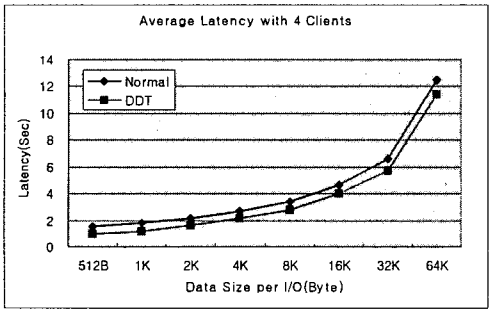


그림 6. Average Latency with 4 Clients

4개의 클라이언트의 경우 DDT 방안을 적용한 경우 0.5~0.9초 정도 Latency가 작았지만 증가폭은 비슷한 양상을 보였다.

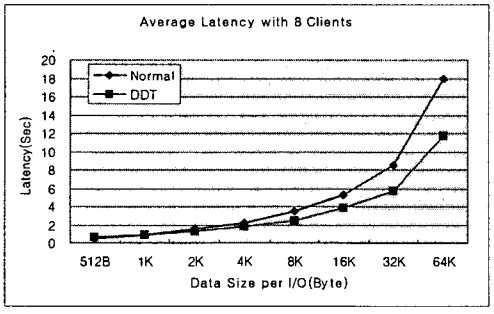


그림 7. Average Latency with 8 Clients

8개의 클라이언트의 경우 64Kbyte에서 DDT방안을 적용하지 않은 경우 Latency가 17.88초로 증가 하였지만 DDT방안을 적용한 경우 4개의 클라이언트 경우와 비슷한 Latency를 보였다. Data Size가 커질수록 그 증가 폭이 차이가 나는 것을 알 수 있다.

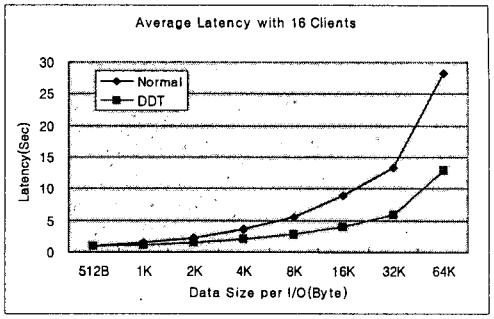


그림 8. Average Latency with 16 Clients

16개의 클라이언트의 경우 Client가 1개일 때와 비교해 Data Size가 64K일 때 Latency가 약 2.5배 증가 하였지만, DDT방안을 적용한 경우 약 1.125배 증가하였을 뿐이다. 이 결과를 통해 클라이언트 수가 증가할수록 서버의 부하로 인해 Latency의 증가폭이 커지는 것을 알 수 있었고 DDT방안을 적용하는 경우 서버 부하가 분산됨을 알 수 있었다.

## 5. 결론 및 향후 연구 방향

본 논문은 iSCSI 기반의 스토리지 서비스 환경에서 서비스 품질을 보장하고 스토리지 서버의 병목 문제를 해길 하기 위한 부하 분산 기법을 제안하였다.

iSCSI 헤더의 예약 필드에 클라이언트 요청 정보를 담아서, 항상 서버를 통해 전달되던 트래픽을 Read Request의 경우 Direct Data Transmission을 통해 요청된 데이터가 스토리지 서버를 거치지 않고 iSCSI Target에서 클라이언트로 직접 전달하도록 한다. 따라서 서버는 Response 트래픽 에 대한 처리 오버헤드를 줄일 수 있다.

실험을 통해 클라이언트 수가 증가됨에 따라 스토리지 서버의 오버헤드가 커지고 그로 인해 Latency가 증가하는 것을 볼 수 있었다. 클라이언트 수가 16개인 경우 약 2.5배의 Latency가 증가하였지만 DDT방 안을 적용한 경우 1.125배의 Latency를 기록하였다. 따라서 DDT방안을 통해 서버의 부하가 효과적으로 분산된다는 것을 알 수 있었다.

향후 연구 과제로는 Data Size 와 서버의 부하 정도에 따라 DDT 적용을 다르게 하여 보다 효과적인 적용 알고리즘에 대한 연구를 진행 하도록 하겠다.

## 6. 참고 문헌

[1] Tom Clark, IP SANs : A Guide to iSCSI, iFCP, and FCIP Protocols for Storage Area Networks, pp.153, Addison-Wesley, 2002.

[2] SAM-3 : Information Technology - SCSI Architecture Model 3, Working Draft, T10 Project 1561-D, Revision7, 9 May, 2003.

[3] Yingping Lu and David H. C. Du, University of Minnesota, "Performance Study of iSCSI-Based Storage Subsystems", IEEE Communication Magazine, pp.76-82, Agust 2003.

[4] 김대근, 박명순 "모바일 디바이스를 위한 iSCSI 기반의 원격 스토리지 서비스에서 중간 서버를 이용한 성능 개선 방안", 정보처리학회논문지, 제11-C권 제6호 pp.843~850, 2004.12

[5] J. Satran: iSCSI Protocol RFC3720, http://www.ietf.org/rfc/rfc3720.txt