DOI: 10.3745/KIPSTC.2009.16C.5.597

대용량 이메일 서비스를 위한 분산 구조 기반의 SMTP 서버

김 영 종[†] · 곽 후 근[††] · 정 규 식[†††]

요 약

SMTP(Simple Mail Transfer Protocol) 서버는 네트워크를 기반으로 사용자의 메일을 전달 해주거나 저장하기 위해 만들어졌다. SMTP 서버는 대규모 서비스를 운영하거나 사용자가 많은 경우 또는 많은 양의 메일을 처리해야 하는 경우에는 SMTP 서버들을 복수로 구성할 수 있어야 한다. SMTP 서버의 메일 저장 공간인 파일시스템을 분리함으로써 복수의 SMTP 서버로 구성하는 것이 가능하며, 이때 분리된 파일시스템을 각각의 SMTP 서버들이 공유하기 위해 NFS(Network File System)를 사용하게 된다. 그러나 NFS는 네트워크 기반의 파일시스템으로써 파일시스템이 가지는 특성을 모두 가지고 있기 때문에, SMTP 서버들이 메일 저장 공간 공유를 위해 사용할 경우 불필요한 작업들로 인해 오버헤드가 발생하게 된다.

본 논문에서는 소켓(Socket)을 통해 직접 작업하는 방식을 사용하여 NFS를 사용함으로 인해 발생하는 작업 오버헤드를 줄이는 방식을 제안한다. NFS를 사용함으로 인해 발생하는 오버헤드를 최소화하기 위해 직접 소켓 작업을 할 수 있도록, 정보 기반의 저장 공간 구조 및 메일 저장을 위한 프로토콜을 정의하였다. 제안된 방식은 Netscape에서 만든 Mailstone을 이용하여 실험을 수행하였고, 실험을 통하여 제안된 방식이 기존 방식에 비해 성능이 향상되었음을 확인하였다.

키워드 : 이메일, SMTP 서버, 메일 서버, 분산 구조, NFS(Network File System)

A Distributed Architecture Based SMTP Server for Large Email Service

Youngjong Kim[†] · Hukeun Kwak[††] · Kyusik Chung[†††]

ABSTRACT

An SMTP(Simple Mail Transfer Protocol) server was designed for delivering and storing user's email across a network. An SMTP server can be distributed as multiple servers for large service, huge users or massive emails. An SMTP server can be constructed by multiple servers with separating file system as email storing space, and each SMTP server can usually share each file system by using the NFS(Network File system). However the NFS is originally designed for sharing each file system across a network, and contains all attributes and features of regular file system. Using this NFS for email storing space of SMTP servers, it makes overhead due to unnecessary work of regular file system.

In this paper, we propose a method to do directly operation with socket for reducing work overhead caused by the NFS. For doing directly operation with socket, this paper defines information based storing space structure and a protocol for storing emails. We performed experiments using Mailstone made by Netscape. The experimental results show the performance improvement of the proposed method compared to the existing method.

Keywords : Email, SMTP Server, Mail Server, Distributed Architecture, NFS(Network File System)

1. 서 론

SMTP(Simple Mail Transfer Protocol)[1, 2] 서버는 네트워크를 기반으로 사용자의 메일을 전달 해주거나 저장하기 위해 만들어진 메일전달 프로토콜을 구현한 서버이다. 메일은 이미 오랜 기간 대중화된 커뮤니케이션 수단으로 사용되어 왔으며 대표적인 SMTP 서버로는 공개소프트웨어인 sendmail[3], qmail[4] 등이 있다. 또한 상용 서버로는 SUN Java System Messaging Server[5], MS Exchange Server[6] 등이 있다.

SMTP 서버의 논리적인 구성은 전달된 메일을 최종 전달지에 전달하기 위한 Client-SMTP와 이를 위해 사용되는 파일시스템 및 Client-SMTP를 사용하는 사용자로 이루어진다. 또한 메일의 최종 전달지 혹은 게이트웨이인 Server-SMTP와 전달된 메일을 저장 혹은 릴레이 해주기 위해 사용되는 파일시스템으로 이루어진다.

대규모 서비스를 운영하거나 사용자가 많은 경우 또는 많은 양의 메일을 처리해야 하는 경우에는 SMTP 서버를 복수로 구성할 수 있어야 한다. 이때 각각의 SMTP 서버는

※ 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌음.
† 정 회 원 : 숭실대학교 정보통신전자공학부 석사과정
†† 정 회 원 : 숭실대학교 정보통신전자공학부 postdoc(교신저자)
††† 정 회 원 : 숭실대학교 정보통신전자공학부 교수
논문접수 : 2008년 11월 17일
수 정 일 : 1차 2009년 3월 20일
심사완료 : 2009년 3월 20일

메일 저장 공간인 파일시스템을 공유할 수 있어야 하며 이를 위해 일반적으로 NFS(Network File System)[7-11]를 사용하게 된다. 각각의 SMTP 서버는 사용자의 메일 저장 공간인 파일시스템을 네트워크 기반으로 마운트 하여 공유하고 이를 통해 SMTP 서버 구성을 확장할 수 있다. 하지만 NFS는 XDR(eXternal Data representation)과 RPC (Remote Procedure Call)를 통해 네트워크를 기반으로 파일과 디렉토리를 공유할 수 있는 분산구조의 파일시스템 구현이 목적이다. 따라서 파일시스템이 가지고 있는 파일 공유를 위한 락킹(Locking) 메커니즘, 계층 구조의 디렉토리 및 작업을 위한 ACL(Access Control List) 등의 파일시스템이 가지고 있어야할 속성들을 가지고 있다. 이 때문에 메일을 저장할 때 불필요한 오버헤드가 발생한다.

이를 보완하기 위해 NAS(Network Attached Storage)[12] Appliance나 SAN(Storage Area Network)[13, 14] Appliance 등의 제품들이 출시되어 있다. 그러나 NAS는 NFS와 유사한 성능문제와 복잡하게 마운트 되어 있는 경우의 관리 등의 문제를 가지고 있고, SAN은 고가의 비용을 필요로 하게 된다. 현재는 서로 다른 파일시스템(NAS와 SAN)을 하나로 묶어 가상의 스토리지로 구성하여 사용할 수 있는 제품들이 출시되고 있다. 그리고 가상의 스토리지내에 있는 각각의 파일시스템들이 성능차이가 있을 경우 부하 분산이 가능하며 관리의 관점에서도 다양한 기능 구현이 가능한 파일 스토리지 가상화(Storage Virtualization)[15, 16]에 관한 연구가 활발히 이루어지고 있다.

본 논문에서는 기존의 SMTP 서버에서 파일시스템을 분리하기위해 NFS를 이용하는 분산 구성이 가지는 문제점을 분석하고, 이를 해결하기 위해 소켓(Socket)을 통해 직접 작업 할 수 있는 구조와 방식을 제안한다. 본 논문의 구성은 다음과 같다. 제 2장에서는 기존의 NFS를 이용한 파일시스템의 분산에 대한 연구방향들과 문제점을 소개한다. 3장에서는 기존 NFS를 이용한 방식의 문제점을 해결하는 새로운 기법을 설명하고, 4장에서는 실험 및 토론을, 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 기존 연구

2.1 연구 동향

기업의 규모가 커져가고 다양한 웹서비스들이 생기면서 기존의 메일 서비스를 확장해 나가야 할 필요성이 증가하고 있고 그 방식도 다양하게 제안되어지고 있다. 확장을 위해서는 SMTP 서버를 복수로 구성할 수 있어야 하고 이때 각각의 SMTP 서버는 메일 저장 공간인 파일시스템을 공유할 수 있어야 한다. 이를 위해 일반적으로 NFS(Network File System)를 사용하게 된다. NFS는 현재 RFC Version 4까지 발표되었지만 분산구조의 파일시스템을 구현하는 것이 목적이라 파일공유를 위한 락킹(Locking) 메커니즘 때문에 발생하는 성능에 대한 논의가 되어 왔다. 그리고 파일시스템의 특성을 가질 필요가 없는 특정 용도의 저장 방법으로 사용할 경우 불필요한 오버헤드가 발생하게 되는 문제점이 있다. 이 때문에 목적에 따른 분산 구조 기반의 파일시스템 성능에 대한 연구들이 활발히 이루어지고 있다.

NFS의 성능 그리고 관리의 어려움 등의 문제로 NAS(Network Attached Storage) Appliance를 사용하는 경우도 있으나 이 또한 마운트 포인트가 늘어남에 따라 발생하는 성능 그리고 관리상의 어려움 등을 구체적으로 해결하지 못하고 있다. 이에 대한 대안으로 SAN(Storage Area Network)이나 iSCSI[17, 18]와 같은 다양한 제품들이 나오고 있고 이제는 이들 제품의 사용이 대중화 되어 있다. 하지만 이는 모두 비용을 추가로 들어야 하거나 하드웨어적인 해결방법으로 경우에 따라 매우 고가의 비용을 필요로 하게 된다.

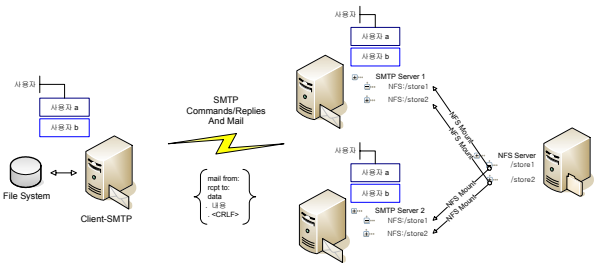
비용을 효과적으로 줄일 수 있는 데이터 관리에 대한 연구는 기업을 중심으로 활발히 이루어져 왔으며 최근에는 NAS, SAN(Storage Area Network), DAS(Direct Attached Storage)[19] 등 스토리지 종류나 프로토콜에 관계없이 하나의 가상의 스토리지로 구성하여 성능 차이가 있는 파일시스템간의 부하 분산 및 이를 하나의 스토리지로 관리할 수 있는 파일 스토리지 가상화(Storage Virtualization)에 관한 연구가 활발히 이루어지고 있다. 현재 출시된 제품으로는 Microsoft의 DFS(Distributed File System)[20]을 기반으로 Brocade의 StorageX Software[21]나 EMC의 Rainfinity Global File Virtualization[22] 등이 있다.

2.2 NFS를 이용한 SMTP 서버

SMTP 서버를 복수로 구성하기 위해서는 각각의 SMTP 서버가 사용자 메일 저장 공간을 공유할 수 있어야 하고 공유된 사용자 메일 저장 공간은 각각의 SMTP 서버 간 작업의 일관성을 유지할 수 있어야 한다. 이를 위해 일반적으로 분산 파일시스템을 구성하기위한 프로토콜인 NFS(Network File System)을 사용한다.

NFS를 이용한 SMTP 서버의 분산 구성은 사용자 메일 저장 공간으로 사용할 공유 디렉토리를 설정한 NFS 서버를 구성하고 각각의 SMTP 서버가 NFS 클라이언트가 되어 NFS 서버에 설정된 공유 디렉토리를 마운트 하여 구성하게 된다. (그림 1)은 NFS를 이용한 SMTP 서버의 분산구성에 관한 그림이다.

NFS 서버는 공유할 디렉토리와 그와 관련된 ACL(Access Control List) 등을 설정하고 접근을 허용한 NFS 클라이언



(그림 1) NFS를 이용한 분산구조

트의 마운트 작업을 기다리게 된다. 분산된 SMTP 서버들은 공유된 메일 저장 공간을 사용하기 위해 NFS 클라이언트가 되어 NFS 서버에 마운트 작업을 요청하여 설정된 공유디렉토리를 사용자의 메일 저장 공간으로 사용하게 되며 이때 NFS 서버는 분산된 SMTP 서버 개수만큼의 마운트 포인트를 가지게 된다.

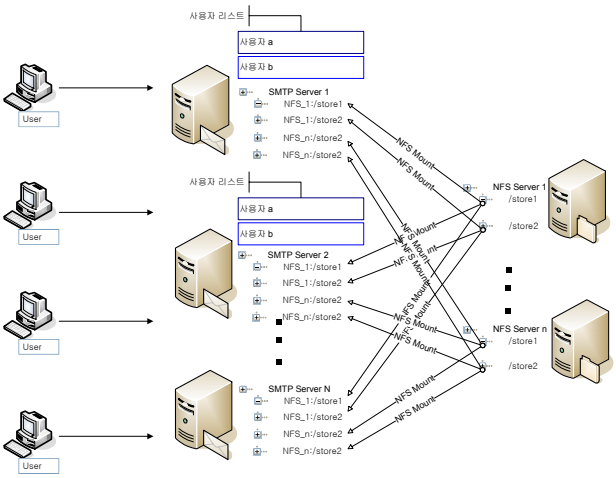
2.3 접근 방식

본 절에서는 기존의 NFS를 사용하여 분산한 SMTP 서버 구조가 가지는 문제점을 정리하고 본 논문의 접근 방식을 간략하게 설명한 후 자세한 내용은 3장에서 기술한다.

2.3.1 NFS를 이용한 SMTP 서버의 문제점

NFS(Network File System)를 통해 SMTP 서버를 분산하였을 경우 NFS 서버의 설정된 공유 디렉토리를 마운트하여 사용자의 메일 저장 공간으로 사용하게 된다. 이때 마운트 포인트인 공유할 디렉토리에 연결할 SMTP 서버 숫자만큼 마운트가 발생하게 되고, 하나의 마운트 포인트에 여러 개의 SMTP 서버가 동시에 접근 할 경우 파일시스템의 특성을 유지하기 위한 프로토콜의 복잡함과 동시 접근을 제어하기 위한 락킹(Locking) 메커니즘으로 인해 성능이 효과적으로 나오지 않는 문제가 있다. 또 구성이 커질수록 복잡하게 구성된 마운트 포인트에 대한 관리 또한 중요한 문제가 된다.

(그림 2)와 같이 확장을 하면 할수록 마운트를 할 디렉토리 숫자만큼 마운트 포인트가 생기게 되고 접속할 SMTP 서버의 숫자만큼 마운트가 이루어져 복잡한 구조를 갖게 된다.



(그림 2) NFS 로 마운트를 하여 구성한 SMTP 서버 구성

2.3.2 본 논문의 접근 방식

본 논문에서는, 각 메일 저장 서버와 소켓(Socket)을 통해 직접 작업하는 방식을 제안한다. 이러한 방식을 사용하면 기존의 NFS를 이용하여 SMTP 서버를 분산구성을 하였을 경우 발생하는 불필요한 오버헤드와 복잡한 구조의 마운트가 이루어지는 것을 없앨 수 있고 이를 통해 성능 저하 현상도 해결 할 수 있다.

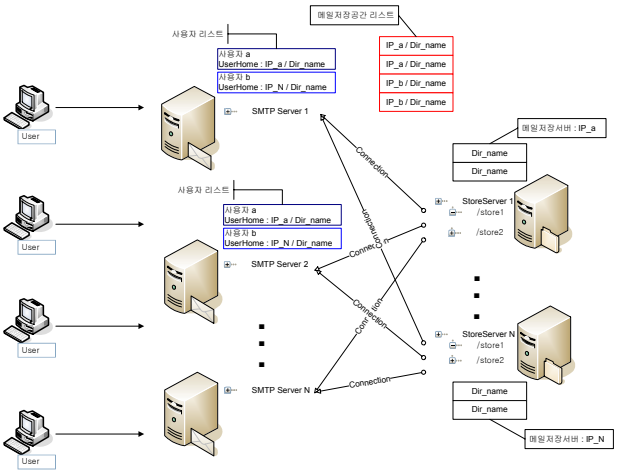
3. 제안된 방식

3.1 전체 구조

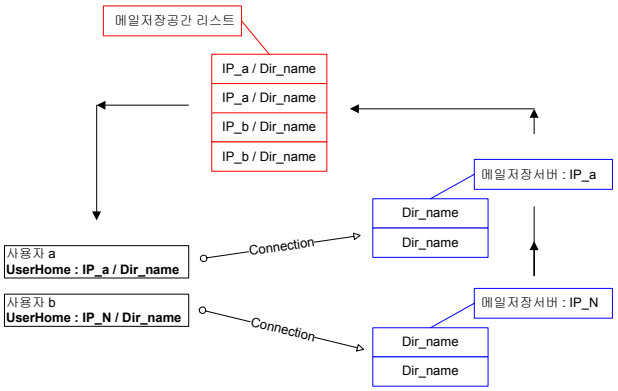
제안된 방식에서는 메일 저장 서버들의 공유할 디렉토리 정보를 IP/Directory_name 형태로 메일 저장 공간 리스트에 저장한다. 사용자 정보에는 UserHome을 추가하여 사용자를 생성할 때 저장된 메일 저장 공간 리스트의 값들을 순차적으로 사용자의 UserHome에 추가를 한 뒤 UserHome 정보로 소켓(Socket)을 통해 직접 작업하는 방식으로 바꾸는 것을 제안한다. (그림 3)은 제안한 방식의 전체적인 그림이다.

제안된 방식에서는 분산 구성을 위해 (그림 4)와 같이 메일 저장 서버의 IP와 메일 저장 공간으로 사용할 디렉토리 이름을 각각의 메일 저장 서버의 정보로 가지고 있고, 각각의 메일 저장 서버의 정보를 메일 저장 공간 리스트에 저장하게 된다. 메일 저장 공간 리스트에는 하나의 정보를 메일저장서버_IP/디렉토리_이름 형태로 보관하게 되며, 각각은 파티션이라는 이름으로 사용되게 된다.

사용자 정보에 UserHome을 추가하고 사용자를 생성할 때 (그림 4)의 메일 저장 공간 리스트의 파티션 정보를 사용자 정보의 UserHome에 할당한다.



(그림 3) 제안된 방식에서의 Operation 경로



(그림 4) 메일 저장 공간 도식도

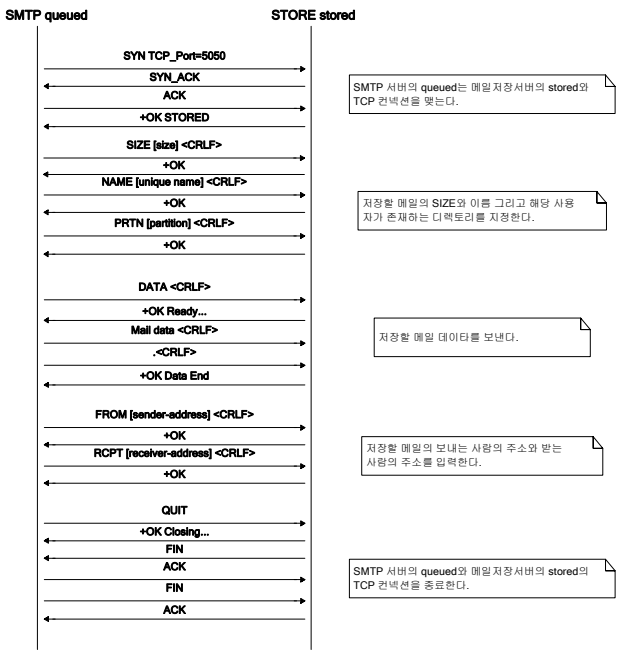
기존에 구현된 큐(queue)를 스캔하여 사용자 메일을 저장하는 queued의 기능을 <표 1>에 정의된 프로토콜을 통해 사용자 메일을 저장하는 형태로 변경하고, <표 1>의 프로토콜을 통해 메일을 저장할 stored 데몬을 구현하였다. queued와 stored는 사용자정보의 UserHome에 있는 파티션 정보를 기반으로 소켓(Socket) 기반의 직접 작업을 통해 메일을 저장하게 된다.

큐(Queue)를 스캔하여 메일을 처리하는 queued와 사용자의 메일을 저장하는 stored 사이의 메일 저장 요청을 처리하기 위해 정의한 메일 저장 프로토콜은 <표 1>과 같다. 본 논문에서는 메일 저장 공간 리스트와 이를 사용한 사용자 정보의 UserHome을 이용하여 분산된 환경에서 메일을 저장하는 queued와 stored가 NFS(Network File System)을 대체하게 된다.

각각의 메일 저장 서버에는 stored 데몬이 실행되고 있어

〈표 1〉 메일 저장 프로토콜

Command	Response	설명
SIZE [size]	+OK -ERR [error string]	저장할 메일 size
NAME [unique name]	+OK -ERR [error string]	저장할 unique name
PRTN [partition]	+OK -ERR [error string]	사용자 정보에 있는 사용자가 위치한 디렉토리 name
DATA	+OK -ERR [error string]	메일 원본 전송
FROM [email address]	+OK -ERR [error string]	보내는 사람의 이메일 주소
RCPT [email address]	+OK -ERR [error string]	받는 사람의 이메일 주소 (여러 명일 경우 반복
QUIT	+OK -ERR [error string]	stored protocol transaction 종료



(그림 5) queued와 stored 프로토콜 시퀀스

SMTP 서버의 queued는 도착한 메일을 저장하기위해 해당 사용자의 정보에서 UserHome을 lookup 한다. 그리고 메일 저장 서버 IP를 통해 사용자가 위치한 메일 저장 서버에 접속을 하고 디렉토리이름을 통해 사용자의 메일 저장 위치를 알아낸 후 정의된 프로토콜을 통해 메일을 저장하게 된다. 이에 대한 구체적인 프로토콜 시퀀스는 (그림 5)와 같다.

3.2 동작 과정

제안된 방식의 구체적인 동작 과정은 (그림 6)과 같다.

단계 a. 메일 저장 서버에서 사용할 자신의 IP와 메일 저장 디렉토리 정보를 메일 저장 서버 별로 저장한다.

단계 b. 모든 메일 저장 서버에 저장된 메일 저장 디렉토리 정보 리스트를 읽어 메일 저장 공간 리스트에 추가한다.

단계 c. 사용자 생성시마다 순차적으로 사용자 정보의 UserHome에 리스트에 있는 메일 저장 서버의 IP/디렉토리 리스트 정보를 넣어준다.

단계 d. smtpd는 외부 SMTP 서버로부터 메일을 받기 위해 일반적인 SMTP 프로토콜 시퀀스를 진행하여 큐(Queue)에 저장하게 된다.

단계 e. queued는 큐를 스캔하여 처리하고자 하는 메일의 SMTP-recipient를 확인한 후 해당 사용자의 사용자 정보 중 UserHome을 읽는다.

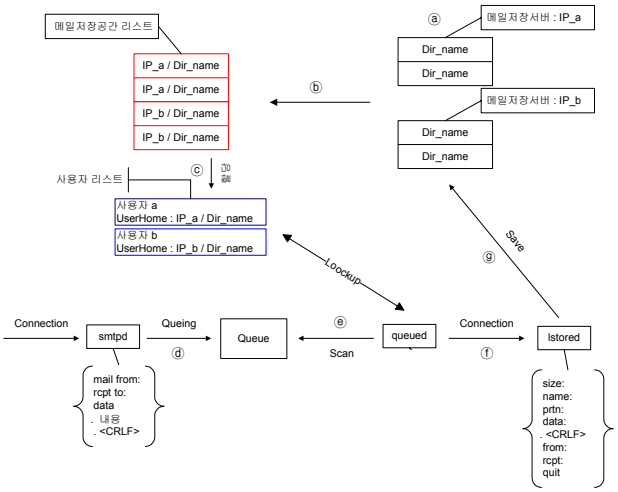
단계 f. UserHome 정보의 서버 IP 이용하여 메일 저장 서버의 stored와 접속한 후 디렉토리 정보를 가지고 정의한 프로토콜 시퀀스에 따라 메일을 저장한다.

단계 g. 메일을 전달하기위해 다른 SMTP 서버로부터 접속이 들어 올 경우 단계 d부터 위와 같은 과정을 반복하게 된다.

3.3 정성적 비교

<표 2>는 기존 방식과 제안된 방식을 구현 방식, 성능, 확장성 관점에서 비교 정리 한 것이다.

기존의 방식에서 사용되는 NFS(Network File System)은 아키텍처와 무관하게 데이터를 표현하기위한 네트워크 기반의 데이터타입을 정의한 XDR(eXternal Data Representation)



(그림 6) 제안된 방식의 동작 과정

〈표 2〉 기존 방식과 제안된 방식의 차이

	구현 방식	성능	확장성
기존의 방식	XDR과 RPC	I/O 발생에 비례 (오버헤드 포함)	제한 있음
제안된 방식	정의된 프로토콜과 이를 기반으로 소켓(Socket)을 통해 직접 작업	I/O 발생에 비례	제한 없음

과 RPC(Remote Procedure Call) 인터페이스를 기반으로 설계되었다. NFS를 사용한 기존의 방식은 각각의 NFS 클라이언트들이 파일을 공유하기 위해 필수적으로 작업 투명성을 보장해야 하며 이를 위해 락킹(Locking) 메커니즘을 사용한다. 또한 작업이 발생하였을 때 계층 구조의 디렉토리와 파일들이 ACL(Access Control List) 등의 속성을 유지해야 함으로 인해 발생하는 오버헤드로 인해 I/O 발생 시 이와 같은 오버헤드를 포함한 I/O에 비례하는 성능을 보이게 된다. 이때 마운트 포인트가 많아지면 많아질수록 오버헤드가 더 발생하게 되어 확장성에도 제한이 발생하게 된다.

안된 방식은 메일을 저장할 때 메일 저장 공간 리스트와 정의한 메일 저장 프로토콜을 구현한 queued와 stored 데몬이 소켓(Socket) 통해 작업을 직접 수행함으로써 I/O 발생 시 실제 발생하는 I/O에 비례하는 성능을 보여주며 메일 저장을 위한 작업만을 수행 하게 되어 효과적인 작업이 가능하고 이로 인해 네트워크 대역폭 내에서 최적화된 확장성 보여준다.

4. 실험 및 토론

4.1 실험 환경

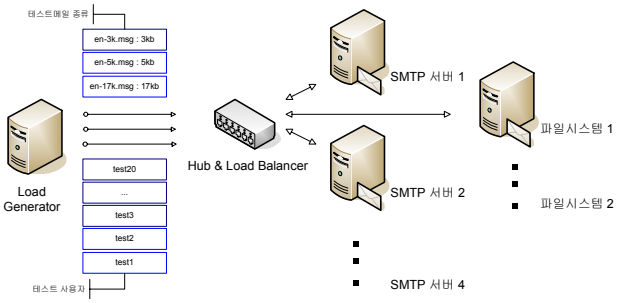
실험은 Netscape에서 만든 메일 테스트 프로그램인 Mailstone[23]을 사용하여 실험을 수행하였다. 실험 구성은 (그림 7)과 같이 메일 트래픽을 발생시키는 Mailstone 1대를 통해 1대의 SMTP 서버와 1대의 파일시스템(1x1), 2대의 SMTP 서버와 1대의 파일시스템(2x1), 4대 SMTP 서버와 1대의 파일시스템(4x1) 및 4대의 SMTP 서버와 2대의 파일시스템(4x2)을 대상으로 실험을 하였다.

실험에 사용된 하드웨어와 설치된 OS(Operating System)는 <표 3>와 같고 실험에 사용한 변수 및 값은 <표 4>와 같다. <표 4>에서 Rampup Time은 테스트 프로그램인 Mailstone이 테스트를 위해 설정한 부하가 발생될 때까지의 준비 시간을 의미한다.

실험에 사용된 메일 사용자는 <표 5>과 같이 총 20개로 사용자 디렉토리가 위치하는 디렉토리인 store1과 store2에 10개씩 고르게 분포되도록 사용자를 생성하였고 실험에 사용된 메일 종류는 <표 6>와 같이 3KB, 5KB, 17KB의 메일을 사용하였다.

Netscape에서 만든 메일 테스트 프로그램인 Mailstone은 <표 5>의 사용자에게 <표 6>의 메일들을 <표 4>의 조건에 따라 주어진 시간만큼 12개의 Client들이 메일을 보내게 된다.

Mailstone은 메일과 관련 프로토콜을 테스트하기 위한 프로그램으로 테스트 가능한 프로토콜로는 SMTP/POP/IMAP 및 웹 메일을 위한 웹 기반 프로토콜이 있다. 테스트 종료 후 남기는 결과 값들은 protocol:aspect:timer_track 포맷으로 보여주며 aspect는 conn, banner, cmd, submit, logout 등이 있으며, timer_track은 시도 횟수 및 에러 횟수 등을 표현하는 Tries, Errors, Bytes written, Bytes read, Time 등이 있다.



(그림 7) 실험의 구성

〈표 3〉 실험에 사용된 하드웨어와 소프트웨어

		Hardware		OS	EA
		CPU	RAM		
SMTP 서버	SMTP 서버	Pentium-R 2.40 GHz	1024MB	Redhat Linux, 2.4.18-3	4
	메일저장서버 (파일시스템)	Pentium-R 2.40 GHz	1024MB	Redhat Linux, 2.4.18-3	2
부하 발생기		Pentium-R 3.00 GHz	512 MB	Redhat Linux, 2.4.18-3	1
부하 분산기		Pumpkin LayerX 2008		N/A	1

〈표 4〉 실험에 사용된 변수 및 값

변수	값
Test Duration	10 minutes
Rampup Time	10 seconds
Number of Clients	12

〈표 5〉 실험에 사용된 사용자

사용자 디렉토리	사용자 계정	용량 제한
$M_HOME/user/store1	test1, test3, test5, test7, test9, test11, test13, test15, test17, test19	N/A
$M_HOME/user/store2	test2, test4, test6, test8, test10, test12, test14, test16, test18, test20	N/A

〈표 6〉 실험에 사용된 메일 종류

파일 명	파일크기	비고
en-3k.msg	3KB	영문
en-5k.msg	5KB	영문
en-17k.msg	17KB	영문

aspect 각각의 의미는 <표 7>과 같다.

본 논문에서는 SMTP 프로토콜을 실험하였고, 결과는 SMTP:aspect:timer_track 형태로 각각의 결과 값이 나온다. 테스트 프로그램의 클라이언트는 connection을 맺은 후 banner를 확인하고 SMTP command를 보낸 뒤 SMTP 프로토콜 시퀀스를 마치고 최종적으로 메시지를 보내는데 성공한 경우 submit 횟수를 추가하며 커넥션을 종료하게 된다. 이에 유효한 테스트 결과 값은 SMTP 프로토콜 시퀀스를 마치고 메시지를 전달하였음을 의미하는 SMTP:submit:Try로 볼 수 있고, 이때 SMTP:submit:Error는 모두 0이어야 한다.

Email은 10분 동안 12개의 client가 3/5/17KB 크기의 메일을 랜덤하게 섞어 보내게 된다. 각각의 테스트 환경별로 email 처리량이 다르기 때문에 email 발생 시간 간격이 서로 다르게 된다. 따라서 email 발생 시간 간격은 각각의 테스트 환경별로 10분/SMTP:submit:try이 된다.

〈표 7〉 Mailstone이 남기는 결과 값의 aspect

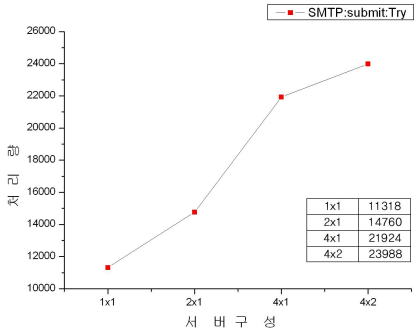
aspect	설명
conn	프로그램이 TCP 컨넥션을 맺은 횟수
banner	커넥션이 맺어진 후 서버가 보여주는 메시지의 횟수
cmd	서버와 클라이언트 사이에 오간 Command 수
submit	cmd 완료 후 메시지를 보낸 횟수
logout	프로그램이 커넥션을 종료한 횟수

4.2 실험 결과

4.2.1 NFS 이용한 SMTP 서버

(그림 8)은 기존 방식으로써 NFS를 이용한 SMTP 서버의 분산 구성을 실험한 결과이다. x 축은 서버의 구성을 나타내고, y 축은 처리량(SMTP:submit:Try)을 나타낸다. 1대일 때의 SMTP:submit:Try 값인 11318을 기준으로, 2x1일 때 약 30%정도의 성능 향상이 이루어 졌고, 4x1일 때는 약 94%의 성능 향상을 보였다. 메일 저장 서버를 1대 더 늘려 4x2로 구성하였을 때는 약 112%의 성능 향상이 있었다.

SMTP 서버를 2대로 늘렸을 때 1대에 비해 약 30% 정도 밖에 성능향상이 되지 않았고, 4대로 늘려 4x1로 구성하였을 때에는 2x1로 구성하였을 때보다 약 48%의 성능 향상을 보였다. 또 메일 저장 서버를 2대로 늘려 4x2로 구성하였으나 역시 4대의 처리량인 21924에 비해 약 10%의 성능 향상 밖에 이루어지지 않았다. 서버의 구성이 증가해도 성능의 향상률이 낮아지는 이유는 NFS에서 발생하는 불필요하는 오버헤드에 기인한다. 이는 마운트 되어진 SMTP 서버 수가 더 늘어나게 되어 메일 저장 서버를 더 늘리더라도 동시 처리량이 더 낮은 곡선을 보이게 될 것임을 예상할 수 있다.

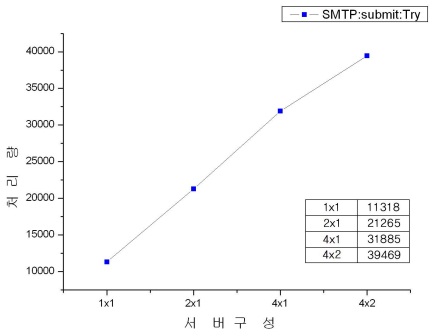


(그림 8) 기존의 방식인 NFS를 이용한 SMTP 분산구성의 처리량

4.2.2 제안된 SMTP 서버

(그림 9)는 제안된 SMTP 서버의 분산 구성을 실험한 결과이다. x 축은 서버의 구성을 나타내고, y 축은 처리량(SMTP:submit:Try)을 나타낸다. 1대일 때의 SMTP:submit:Try 값인 11318을 기준으로, 2x1일 때 약 88%의 향상이 있었고, 4x1일 때 약 182%의 향상이 있었다. 메일 저장 서버를 1대 더 늘려 4x2로 구성하였을 때는 약 248%의 성능향상이 있었다.

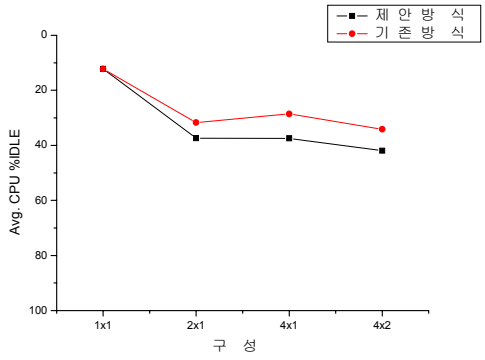
SMTP 서버를 2대로 늘려서 구성하였을 때 1대에 비해 약 88% 향상됨을 보여주고 있다. 4x1로 구성하였을 때는 2x1에 비해 약 50%의 향상을 보여주고 있다. 2x1이 1대에 비해 약 2배 가까이 성능향상이 있었으나 4x1로 구성하였을 때는 2x1에 비해 50%정도의 향상이 되었음은 메일 저장 서버에 효과적인 부하가 발생함을 의미한다. 이에 메일 저장 서버를 한 대 더 늘려 4x2로 구성하여 테스트한 결과 2x1에 비해서는 약 85%의 성능향상이 있었고 4x1에 비해 약 24%의 성능 향상을 보이고 있다. 이러한 성능 향상의 원인은 제안된 방법이 효과적으로 대용량의 메일을 처리함을 의미한다. 이는 동시 처리량과 구성한 각각의 서버의 성능에 따라 적당한 배치가 가능하며 이를 통해 효과적으로 자원을 사용하여 최적의 성능을 위한 구성이 가능하다는 것을 예상할 수 있다.



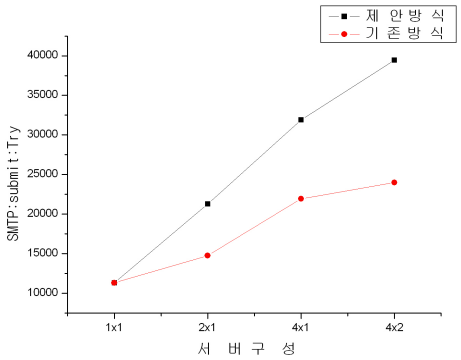
(그림 9) 제안된 방식의 SMTP 분산구성의 처리량

4.2.3 CPU 사용량

(그림 10)은 기존의 방식인 NFS를 사용하였을 때와 제안된 방식을 사용하였을 때의 CPU의 평균 IDLE 시간을 나타낸다. 그림에서 x 축은 본 논문에서 실험한 구성을 나타내



(그림 10) 기존 방식과 제안된 방식의 평균 CPU 사용량



(그림 11) 제안방식이 기존방식보다 좋아지는 처리량

고, y 축은 CPU의 평균 IDLE 시간을 나타낸다. 1대 구성인 1x1의 IDLE 시간을 기준으로 각각의 실험의 구성에서 기존 NFS를 사용한 방법과 제안된 방식을 비교했을 때 평균적으로 5%의 차이가 남을 알 수 있다.

이는 제안된 방법이 기존의 NFS를 이용한 구성 방법에 비해 시스템의 자원(CPU)을 평균 5% 정도를 절약함을 의미하고, 1대의 실험결과와 2x1 구성, 4x1 구성을 각각 비교해 볼 때 실험의 구성이 늘어날수록(실험에 사용된 서버의 대수가 커질수록) 이 수치는 더 커질 것으로 예상된다. 기존의 NFS를 이용한 방식과 제안된 방식의 처리량에 대한 정량적 비교 분석을 해볼 경우 CPU의 평균 IDLE 시간은 좀 더 의미가 갖게 된다. 즉 제안된 구조는 기존 구조에 비해 확장성(처리량)이 높은 반면 시스템 자원(CPU)은 덜 사용한다.

2.4.4 정량적 비교

(그림 11)는 기존의 방식인 NFS를 사용하였을 때와 제안된 방식을 사용하였을 때 각각을 서버 구성을 변경하여 실험한 결과를 비교하였다.

기존의 방식인 NFS를 사용하였을 때는 서버가 늘어남에 따라 향상되는 성능이 1대의 처리량을 기준으로 할 때 각각 약 30%, 약 94%, 약 112% 임에 비해 제안된 방식을 사용할 경우 약 88%, 약 182%, 약 249%의 효과적인 성능 향상이 나타남을 보여준다.

<표 8>의 4x2 구성의 값을 비교해볼 때 기존 방식인 NFS의 성능향상이 1대일 때의 SMTP:submit:Try 값을 기준으로 하여 약 94%에서 약 112%로 성능향상을 보였으나 제안된 방식의 경우 약 182%에서 약 249%로 매우 효과적인 성능향상이 이루어 졌음을 볼 수 있다. 이는 기존의 방식은 부하를 분산시키기 위해 메일 저장 서버를 늘려도 큰 효과가 없지만, 제안된 방식은 오버헤드 없이 네트워크를 사용하여 부하 분산이 효과적으로 이루어져 최적화된 성능 향상이 나타남을 보여준다.

〈표 8〉 기존방식과 제안방식의 처리량

구성	SMTP:submit:Try : 기존방식	SMTP:submit:Try : 제안방식
1x1	11318	11318
2x1	14760	21265
4x1	21924	31885
4x2	23988	39469

4.3 토론

기존의 NFS(Network File System)를 사용한 SMTP 분산 구성 방식은 다음의 3가지 이유로 인해 메일을 저장할 때 불필요한 오버헤드가 발생한다. 첫 번째는 NFS가 파일을 공유하기 위해 사용하는 락킹(Locking) 메커니즘 때문에 발생하고, 두 번째는 파일시스템이 가지고 있는 계층 구조의 디렉토리를 처리하기 위한 구조로 인해 발생하고, 마지막으로 세 번째는 작업을 위한 ACL(Access Control List) 등의 속성을 처리하면서 발생한다.

본 논문에서 제안된 방식은 대량의 메일을 처리하기위해 SMTP 서버를 분산시켜야 할 경우 메일 저장을 위해 필요한 작업만 소켓(Socket) 을 통해 직접 수행함으로써 불필요한 오버헤드를 없애 처리 성능을 효과적으로 높일 수 있다. 제안된 방식은 NFS에서 발생하는 오버헤드를 최소화하기 위해 직접 소켓을 통해 작업을 할 수 있도록, 정보 기반의 저장 공간 구조와 메일 저장을 위한 프로토콜을 정의하였다.

본 논문의 제안된 방식의 단점은 비록 내부 트래픽이기는 하나 오가는 데이터가 보안에 취약할 수 있다는 것이고, 이는 SSL 등 보안관련 라이브러리들을 적용하는 것으로 해결할 수 있다.

5. 결 론

본 논문에서는 기존 NFS 통해 메일 저장 공간을 공유하는 방식에서 직접 소켓(Socket)을 통해 작업을 할 수 있도록, 정보기반의 저장 공간 구조와 메일 저장을 위한 프로토콜을 정의하여 이를 통해 메일을 저장하는 방식을 제안하였다. 기존의 NFS를 이용한 방식이 작업 오버헤드가 발생하는 것에 비해서, 제안된 방식은 직접 소켓을 통해 작업을 수행하여 작업 오버헤드를 줄일 수 있었다. 제안된 방식은 실험을 통하여 기존의 NFS를 통해 메일 저장 공간을 공유하는 방식보다 메일 처리량이 증가되었음과 CPU 자원을 효과적으로 사용함을 확인하였다. 이는 불필요한 작업을 제거

함으로써 처리할 수 있는 메일 트래픽양이 늘어났음을 의미한다.

향후 연구 방향으로는, 좀 더 효과적인 메일 트래픽 처리를 위해 큐(Queue) 작업을 최적화하는 것이다. 즉, 해당 프로세스들이 들어온 메일을 처리하기 위해 큐를 스캔(Scan)하는데 분리되어있는 각각의 큐에 담당 프로세스를 할당하여 스캔속도를 높여 처리속도를 향상시킬 것이다. 그리고 SMTP 서버와 메일 저장 서버의 정보를 더 세분화하여 관리적인 관점에서도 효율을 높일 것이다.

참 고 문 헌

[1] J. Postel, "Simple Mail Transfer Protocol", IETF RFC 821, 1982.

[2] J. Klensin, "Simple Mail Transfer Protocol", IETF RFC 2821, 2001.

[3] sendmail, http://www.sendmail.org/

[4] qmail, http://www.qmail.org/

[5] SUN Java System Messaging Server, http://www.sun.com/software/products/ messaging_srvr/index.xml

[6] MS Exchange Server, http://www.microsoft.com/EXCHANGE/default.mspx

[7] Sun Microsystem Inc, "NFS: Network File System Protocol Specification", IETF RFC 1094, 1989.

[8] Sun Microsystem Inc, "NFS Version 3 Protocol Specification", IETF RFC 1813, 1995.

[9] S. Shpler, B. Callaghen, D. Robinson, R. Thurlow, C. Beame, M.Eisler, and D. Noveck, "Network File System (NFS) version 4 Protocol", IETF RFC 3010, 2000.

[10] RPC, http://en.wikipedia.org/wiki/Remote_procedure_call

[11] XDR, http://en.wikipedia.org/wiki/External_Data_Representation

[12] B. Reed, E. Chron, R. Burns, and D. Long, "Authenticating network attached storage", IEEE Micro, Vol.20, No.1, pp. 49-57, 2000.

[13] S. Yin, Y. Luo, L. Zong, S. Rago, J. Yu, N. Ansari, and T. Wang, "Storage area network extension over passive optical networks (S-PONS)", IEEE Communication Magazine, Vol. 46, No.1, pp.44-52, 2008.

[14] G. Zhang, J. Shu, W. Xue, and W. Zheng, "Design and Implementation of an Out-of-Band Virtualization System for Large SANs", IEEE Transactions on Computers, Vol.56, No. 12, pp.1654-1665, 2007.

[15] N. Leavitt, "Application Awareness Makes Storage More Useful", IEEE Computer, Vol.41, No.7, pp.11-13, 2008.

[16] M. Kallahalla, M. Uysal, R. Swaminathan, D. Lowell, M. Wray, T. Christian, N. Edwards, C. Dalton, and F. Gittler, "SoftUDC: a software-based data center for utility computing", IEEE Computer, Vol.37, No.11, pp.38-46, 2004.

[17] L. Yingping and D. Du, "Performance study of iSCSI-based storage subsystems", IEEE Communications Magazine, Vol. 41, No.8, pp.76-82, 2003.

[18] K. Meth and J. Satran, "Features of the iSCSI protocol", IEEE Communications Magazine, Vol.41, No.8, pp.72-75, 2003.

[19] DAS, http://en.wikipedia.org/wiki/Direct-attached_storage.

[20] DFS, http://en.wikipedia.org/wiki/Distributed_file_system.

[21] Brocade, http://www.brocade.com/products-solutions/products/file-management /product-details/storagex/index.page

[22] EMC, http://www.emc.com/solutions/business-need/virtualizing-information- infrastructure/file-virtualizations.htm

[23] Mailstone, http://docs.sun.com/source/816-6036-10/index.html

김 영 종

e-mail : opensys@q.ssu.ac.kr

1996년 7월~1998년 4월 ㈜한글과컴퓨터연구원

2000년 9월~2004년 11월 ㈜캐스트와이즈 대표이사

2006년 1월~2008년 4월 ㈜하우리/대표이사

2007년 7월~2009년 3월 열린사이버대학교/정보지원처장

2007년 12월~2009년 03월 오픈소스커뮤니티연구소/소장

2007년 3월~현 재 숭실대학교 정보통신전자공학부 석사과정

관심분야 : 네트워크 컴퓨팅 및 보안

곽 후 근

e-mail : gobarian@q.ssu.ac.kr

1996년 호서대학교 전자공학과(학사)

1998년 숭실대학교 전자공학과(석사)

1998년~2006 숭실대학교 전자공학과(박사)

1998년 8월~2000년 7월 (주) 3R 부설 연구소 주임 연구원

2006년 3월~현 재 숭실대학교 정보통신전자공학부 postdoc

관심분야 : 네트워크 컴퓨팅 및 보안

정 규 식

e-mail : kchung@q.ssu.ac.kr

1979년 서울대학교 전자공학과(공학사)

1981년 한국과학기술원 전산학과(이학석사)

1986년 미국 University of Southern California (컴퓨터공학석사)

1990년 미국 University of Southern California (컴퓨터공학박사)

1998년 2월~1999년 2월 미국 IBM Almaden 연구소 방문 연구원

1990년 9월~현 재 숭실대학교 정보통신전자공학부 교수

관심분야 : 네트워크 컴퓨팅 및 보안