SAN 환경에서 공유파일시스템을 위한 Fault-tolerant server 구현

*최 영 한, 이 주 평, 이 철, 박 규 호
한국과학기술원 전자전산학과
전화 : 042-869-5425 / 핸드폰 : 016-803-4212

Design and Implementation of Fault-tolerant server for Shared File System in SAN environment

*Young-Han Choi, Ju-Pyung Lee, Chul Lee, Kyu-ho Park
Dept. of Electrical Engineering & Computer Science, KAIST
E-mail : yhchoi@core.kaist.ac.kr

Abstract

This paper designs and implements fault-tolerant server of meta server for shared filesystem (SANfs) in SAN environment. SANfs[1] is the filesystem that many clients can share data in Network-attached stroage in SAN environment and meta server is the server that processes file operation in SANfs.

The focus of this paper is the implementation of fault-tolerant server of meta server in SANfs. In the event of a meta server, meta server failovers to a fault-tolerant server where its processing continues seamlessly. If meta server doesn't restore, fault-tolerant server searches reliable client and makes another fault-tolerant server and work as meta server. Heartbeat monitors meta server and shadow server and controls them.

I. 서론

Network-attached storage의 도움으로 스토리지는 로컬 컴퓨터만 접근 할 수 있다는 제한에서 벗어나 여러 클라이언트들이 접근이 가능하게 되었다. 클라이언트들의 동시 접근으로 인해 파일의 inode 관리, lock 관리등 파일의 무결성(integration)을 유지할 수 있는 파일시스템이 요구되었고 이것을 충족시키는 파일 시스템인 공유파일시스템(shared filesystem)이 나오게 되었다. 파일을 관리하기 위해 파일매니저(file manager)가 필요한데 이것을 하나의 서버가 전담하여 관리하는 central server형, 여러 컴퓨터가 파일 관리일을 하는 distributed형, 마지막으로 클라이언트와 서버가 일을 분담하여 하는 private/merged형으로 나누고 있다. 각각에 해당하는 파일 시스템으로 NFS[2], AFS[3], GFS[4]가 있다.

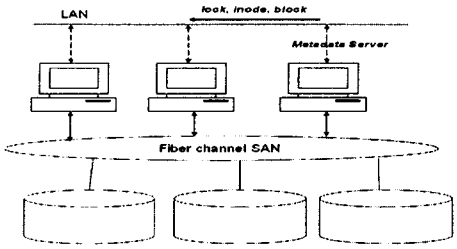
SANfs[1]는 central server형 공유 파일시스템으로 전체 파일을 관리하는 meta server를 두고 있다. 중앙 서버가 있으면 관리면에서 효율성이 있으나 그 서버가 fault가 나게 되면 전체 시스템에 영향을 주어 동작을 멈추게 하는 문제가 있다. 본 논문에서는 위의 문제를 해결할 수 있는 fault-tolerant 서버를 제안, 구현하였고 shadow server로 명명하였다.

본 논문에서 제시한 shadow 서버는 failover시에도 지속적으로 서비스를 할 수 있으며 meta server가 오랬동안 회복되지 않는 상황에서도 SANfs의 reliability를 유지시키기 위해 shadow server의 기능을 클라이언트에게 전이 할 수 있는 기능도 구현하였다.

II. SANfs

2.1 SANfs[1] 기본 구조

SANfs는 Linux 기반의 공유 파일 시스템이며 linux kernal 상에서 개발되었다. SANfs는 그 데이터의 일관성을 유지하기 위한 file manager를 Meta Server라 부른다. 파일 서비스를 요청하는 호스트들과 메타 서버와는 서로 100MBps 이더넷으로 연결되어 있으며, 각 호스트와 메타 서버간의 파일 관리메세지는 TCP 통신을 한다. 그림 1.1은 리눅스 커널 상에서 SANfs의 구조를 나타낸 것이다.



(그림 1.1) SANfs의 구조

데이터가 저장되는 디스크들과 호스트들, meta server는 Fiber Channel로 연결된 Storage Area Network (SAN)을 형성하며, 호스트들은 디스크로 직접 억세스를 한다. SANfs 공유 파일 시스템 클라이언트는 리눅스의 VFS(Virtual File System)의 아래에 존재하는 여러 파일시스템 중의 하나로 등록된다.

2.2 SANfs의 한계점

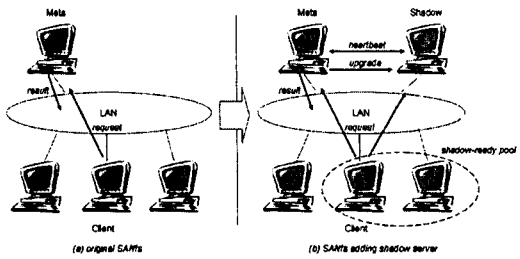
SANfs는 central server형 공유 파일 시스템이기 때문에 파일은 meta server에서 집중적으로 관리를 하게 된다. 따라서 meta server에서 fault가 나게 되면 SANfs는 더 이상의 서비스를 제공할 수 없는 치명적인 문제인 가지게 된다.. 그러기에 meta server가 더 이상 서비스를 할 수 없는 경우 그 기능을 대신할 서버가 필요하게 된다. 본 논문에서는 이 일을 담당할 수 있는 fault-tolerant 서버, 즉 shadow server를 구현하였다. 그리고 Heartbeat를 두어 meta server와 shadow server를 감시하도록 하였으며 각 서버가 SANfs에서 정상적으로 작동하도록 컨트롤(control)의 기능을 첨가하였다.

III. Shadow Server의 설계 및 구현

3.1 shadow server가 첨가된 SANfs의 전체구조

(1) 전체구조

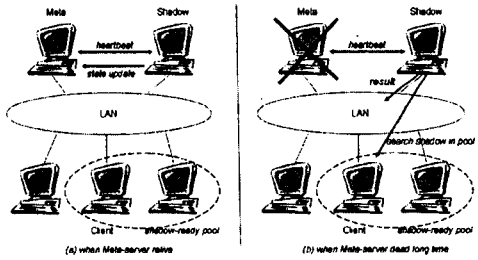
Shadow server가 첨가된 SANfs의 전체구조는 (그림 3.1)과 같다. 클라이언트는 파일 서비스를 요청하기 위하여 meta server와 shadow server에 동시에 두개의 connection을 맺고 request를 보내게 된다[5]. Meta server에서는 request을 가지고 파일에 대한 서비스를 해 주게 되며 해당 서비스의 결과를 클라이언트와 shadow server에게 보내준다.



(그림 3.1) 전체 구조

shadow 서버는 클라이언트에게 받은 request와 meta server에서 받은 결과를 가지고 file에 관한 상태를 업데이트 시킨다. Meta server와 shadow server가 서로 살아 있음은 heartbeat를 통해서 이루어 진다. 그리고 두 서버가 죽을 수 있는 최악의 경우를 대비해 클라이언트 중 몇 개를 묶어 예비 shadow server를 사용할 수 있도록 shadow-ready pool을 이루고 heartbeat에서 관리한다.

(2) Meta server의 fault



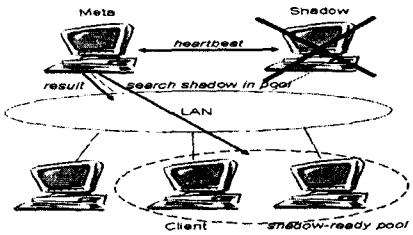
(그림 3-2) Meta server가 fault일 경우

meta server가 fault가 났을 경우는 (그림 3-2)과 같이 두가지 이다. 하나는 Meta server가 fault가 난 후 어느 정도의 시간이 흐른 후 재기능을 할 수 있을 경우이며, 다른 하나는 meta server가 지속적으로 서비스를 할 수 없는 경우이다.

첫 번째 경우에 문제가 되는 것은 meta server가 서비스를 다시 시작하기 위해 그동안의 상태 정보를 알지 못한다는 것이다. 이때는 클라이언트로부터 서비스의 요청이 적은 시점에 shadow server의 서비스를 blocking 시키고 meta server의 상태를 업데이트 시키도록 하였다. 두 번째 경우는 shadow server 역시 fault 나는 경우를 배제할 수 없으므로 shadow-ready pool에서 적당한 클라이언트를 골라 shadow server로 지정하고 자신의 상태를 업그레이드 시켜 shadow

server로 작동시키도록 한다.

(3) Shadow server의 fault

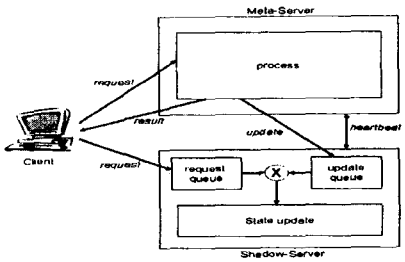


(그림 3-3) Shadow server가 fault일 경우

Shadow server가 fault일 경우는 (그림 3-3)과 같다. Meta server가 shadow server의 fault를 알았을 경우 shadow-ready pool에서 적당한 클라이언트를 골라 자신의 파일 정보를 업데이트 시키고 안정적으로 두개의 서버로 SANfs에서 서비스를 해 주도록 한다.

3.2 Shadow server의 구조

(1) Shadow server의 전체구조

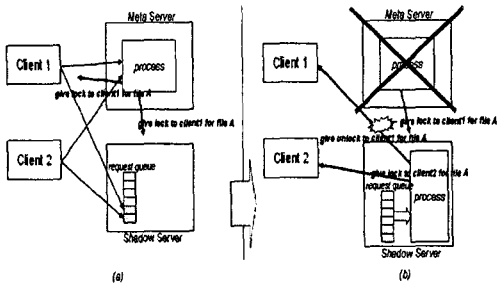


(그림 3-4) Shadow server의 구조

Shadow server의 전체구조는 (그림 3-4)와 같다. Shadow server는 두개의 queue를 가지고 있다. 하나는 클라이언트에서 오는 request를 저장하는 request queue, 그리고 다른 하나는 meta server에서 오는 결과를 저장하는 update queue이다. 각각의 queue에 request와 result가 저장이 되면 일치하는 두개의 데이터를 가지고 shadow server는 file에 대한 정보를 업데이트 시킨다.

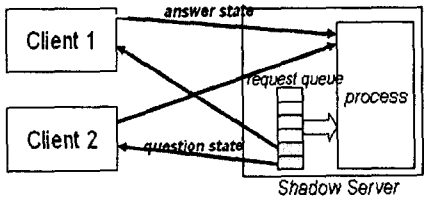
(2) Meta server와 Shadow server의 동기화

file의 정보를 업데이트 시킬때 가장 중요한 것은 meta server와 shadow server의 상태의 동기를 맞추는 것이다.



(그림 3-5) 동기화 문제 발생

(그림 3-5)와 같이 meta server에서 업그레이드된 정보를 클라이언트에게만 전달하고 shadow server에 제대로 전달을 시키지 않았을 경우 shadow server는 잘못된 결과를 가지고 서비스를 해 줄 수 있는 경우가 생긴다. 이와 같은 결과는 shadow server가 meta의 진행상황을 result가 도착하기 전에 제대로 파악을 할 수 없기 때문이다.



(그림 3-6) 동기화 문제 해결

이 문제를 해결하기 위해서 shadow server는 request queue에서 서비스를 해 주지 않은 request들을 클라이언트에게 request 처리 유무 및 결과에 대한 정보를 요청하고 받은 결과를 가지고 파일의 정보를 업데이트 시킴으로써 파일 동기화 문제를 해결하였다.

또 이렇게 함으로써 shadow server는 서비스 대기중인 file의 정보를 알 수 있기 때문에 대기 중인 파일 이외에 대해서는 blocking 없이 끊임없이 서비스를 해 줄 수 있다.

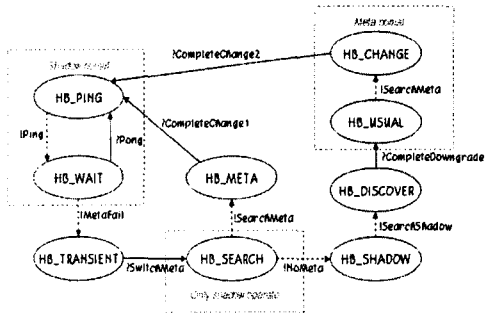
3.3 Heartbeat

Meta server와 shadow server가 서로 살아 있음을 확인하기 위해 주기적으로 신호를 교환해야 한다. 이 일을 하는 것이 heartbeat이다.

heartbeat는 주기적으로 신호를 보내고 있으며 서로 살아 있음을 확인만 하기 때문에 UDP을 통해 구현하

였다.

그리고 전체 시스템의 관리 및 shadow-ready pool에서의 예비 shadow server의 찾기 등 SANfs가 정상적으로 서비스하기 위한 컨트롤(Control)을 담당한다.



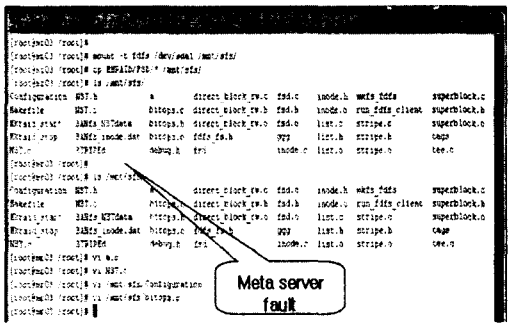
(그림 3-7) heartbeat의 state diagram

(그림 3-7)과 같이 heartbeat는 전체 시스템을 모니터링하고 있다가 각 상황에 맞는 메시지를 shadow server에 보내게 되며 SANfs가 적절한 서비스를 제공할 수 있도록 컨트롤하는 역할을 한다.

Ⅳ. Shadow server의 실험

3장에서 설계한 shadow server를 구현하였다. Meta server, shadow server 그리고 클라이언트 총 3대의 컴퓨터를 사용하였다.

(그림 4-1)은 meta server가 fault일 경우에도 SANfs가 정상적으로 작동함을 보여 주고 있다. /mnt/sfs에 SANfs를 mount 시켰으며 meta server가 fault 난 이후에도 shadow server에서 지속적으로 /mnt/sfs 디렉토리에 접근을 하여 파일에 대한 서비스를 받고 있음을 알 수 있다.



(그림 4-1) 결과

Ⅴ. 결론

공유 파일 시스템으로 만들어진 SANfs에서 meta server의 fault는 전체 시스템에 큰 영향을 미친다.

본 논문에서는 meta server의 fault시 failover 하여 지속적으로 SANfs를 작동시킬 수 있는 fault-tolerant server인 shadow server를 설계 및 구현하였다. Shadow server는 failover시에 일어날 수 있는 동기화 문제를 해결하였다. 그리고 failover시 request queue를 사용하여 지속적으로 파일에 대한 서비스를 할 수 있도록 하였다.

본 논문에서 사용한 구조를 이용하여 서버의 fault시에도 지속적으로 서비스를 할 수 있도록 하는데 확장할 수 것으로 사료된다.

참고문헌

[1] 황주영, "SAN 기반 고성능 공유 파일 시스템", 2003. 2.

[2] P. Staubach, C. Smith, D. Lebel, B. Pawlowski, C. Juszcak and D. Hitz, "Nfs version 3: Design and implementation", 1994. 6

[3] P. Kumar, M. E. Okasaki, E. H. Siegel, M. Satyanarayanan, J. J. Kistler and D. C. Steere, "Coda: A highly available file system for a distributed workstation environment", 1990

[4] S. Soltis, G. Erickson, K. Preslan, M. O'Keefe, and T. Ruwart, "The global file system: A file system for shared disk storage", 1997

[5] Noel Burton-Krahn, "HotSwap - Transparent Server Failover for Linux", 2002