SAN 기반 공유 파일 시스템에서 Fault-tolerance를 위한 Shadow Server 구현

최영한°, 김형천, 홍순좌
국가보안기술연구소
{yhchoi°, khche, hongsj}@etri.re.kr

Implementation of Shadow Server for Fault-tolerance in SAN-based Shared File System

Young Han, Choi° Hyoungchun Kim, Soonjwa Hong
National Security Research Institute

요 약

본 논문에서는 SAN 기반 공유 파일 시스템인 SANfs의 fault-tolerance를 보장 받기 위해 fault-tolerant server인 shadow server를 구현하였다. SANfs[1]는 SAN에서 Network-attached storage에 접근하는 여러 클라이언트가 서로의 데이터를 공유할 수 있도록 도와주는 파일시스템이다. SANfs에서 파일 관리를 위해 meta server를 두고 있으며, 이 서버에서 네트워크를 통해 접근하는 여러 클라이언트의 request를 관리한다. SANfs에서는 meta server를 통해 중앙 집중식으로 파일시스템을 관리하고 있기에 meta server가 fault가 나게 되면 전체 시스템의 동작이 멈추게 되는 single point-of-failure의 문제가 생기게 된다. 본 논문에서는 meta server가 fault가 났을 경우에도 지속적으로 서비스를 할 수 있도록 shadow server를 두었으며, 이 서버가 meta server의 이상 시 그 기능을 대행하도록 하였다. 본 논문의 shadow server는 평상시에 meta server와 파일시스템의 metadata의 동기를 맞추고 있으며, 이 정보를 가지고 meta server로 그 기능을 전환 하였을 때 서비스를 해 주도록 하고 있다. 상대 서버의 이상 유무의 판단은 heartbeat를 통해 이루어지고 있으며, meta server로의 failover는 heartbeat의 주기에 영향을 받음을 실험을 통해 알게 되었다.

1. 서 론

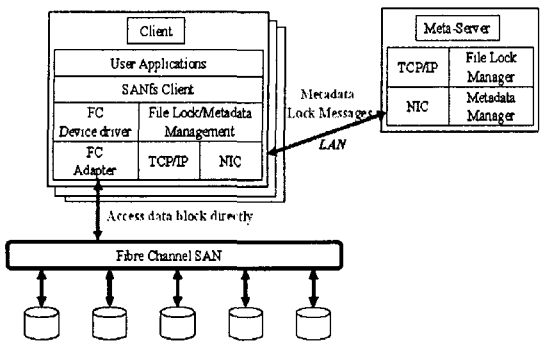
Network-attached storage의 도움으로 스토리지는 로컬 컴퓨터만 접근 할 수 있다는 제한에서 벗어나 여러 클라이언트들도 접근이 가능하게 되었다. 클라이언트들의 동시 접근으로 인해 파일의 inode 관리, lock 관리등 파일의 무결성(integration)을 유지할 수 있는 파일시스템이 요구되었고 이것을 충족시키는 파일 시스템인 공유파일시스템(shared filesystem)이 나오게 되었다[7]. 파일을 관리하기 위해 파일매니저(file manager)가 필요한데 이것을 하나의 서버가 전담하여 관리하는 central server형, 여러 컴퓨터가 파일 관리를 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 관리 면에서 효율성이 있으나 그 서버가 fault가 나게 되면 전체 시스템에 영향을 주는 single point-of-failure가 생기는 문제가 있다. 본 논문에서는 위의 문제를 해결할 수 있는 fault-tolerant server를 제안, 구현하였고 shadow server로 명명하였다.

2장에서는 SANfs 구조 및 한계에 대해 설명을 하겠고, 3장에서는 본 논문에서 구현한 shadow server에 대한 구조 및 동작 메커니즘를 설명하겠다. 4장에서는 shadow server의 성능 측정을 하였으며 마지막으로 결론을 맺고 있다.

2. SANfs의 구조 및 한계

SANfs[1]는 리눅스 기반의 공유 파일 시스템이며 리눅스 커널 상에서 개발되었다. SANfs는 그 데이터의 일관성을 유지하기 위한 file manager를 두고 있으며, 그것을 meta server라 부른다. 파일 서비스를 요청하는 호스트들과 메타 서버와는 서로 100MBps 이더넷(Ethernet)으로 연결되어 있으며, 각 호스트와 메타 서버간의 파일 관리 메세지는 TCP 통신을 한다. [그림 1]은 리눅스 커널 상에서 SANfs의 구조를 나타낸 것이다.



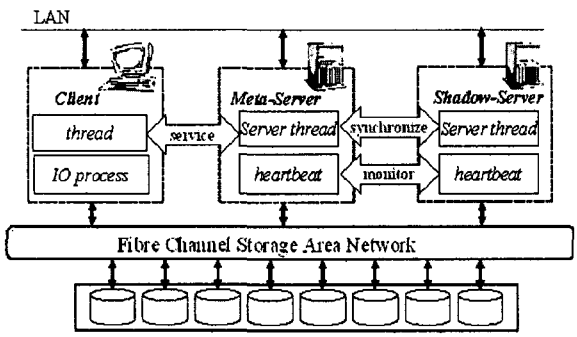
[그림 1] SANfs의 구조

SANfs는 central server형 공유 파일 시스템이기 때문에 파일은 meta server에서 파일 시스템에 대한 정보를 집중적으로 관리한다. 따라서 meta server에서 fault가 나게 되면 SANfs는 더 이상의 서비스를 제공할 수 없는 single point-of-failure 문제가 생기게 된다. 그래서 meta server가 더 이상의 서비스를 제공할 수 없는 경우 meta server의 기능을 하는 또 다른 서버가 요구된다. 그래서 본 논문에서는 meta server가 fault가 났을 경우 그 기능을 대신할 수 있는 shadow server를 구현하였다.

3. Shadow Server의 설계 및 구현

3.1 Shadow server가 첨가된 SANfs의 전체구조

Shadow server가 첨가된 SANfs의 전체구조를 보면 [그림 2]와 같다.

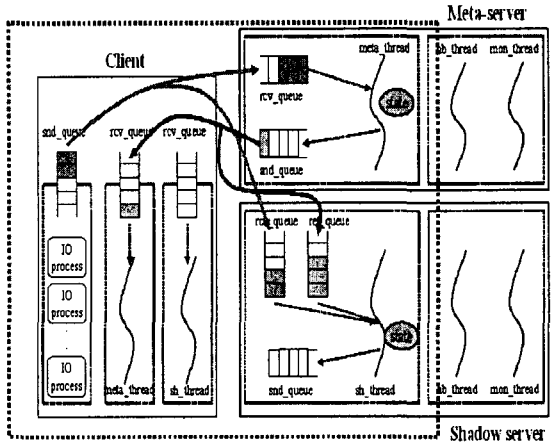


[그림 2] Shadow server가 첨가된 SANfs의 전체 구조

Meta server와 shadow server는 자신의 기능을 수행하는 thread를 가지고 있다. 각 서버가 작동을 하면서 고려해야 할 중요한 문제는 자신이 가지고 있는 file의 metadata가 서로 동일해야 한다는 것이다. meta server와 shadow server는 metadata가 변경될 때마다 서로 동기화를 맞추고 있다. 그리고 서버가 이상이 있는지 heartbeat를 통해서 주기적으로 monitoring하고 있다. Heartbeat에서 자신이 살아 있음을 주기적으로 신호를 보내는 부분과 상대 서버의 신호를 받아서 이상 유무를 판단하는 부분인 두 개의 thread가 작동을 하고 있다. 클라이언트에서는 meta server와 shadow server에 두 개의 connection을 맺고 있으며 정상적인 SANfs에서는 meta server와 연결된 connection에서 서비스를 받으며, meta server가 fail이 났을 경우에는 shadow server에 연결된 connection을 통해서 서비스를 받는다.

3.2 작동 메커니즘

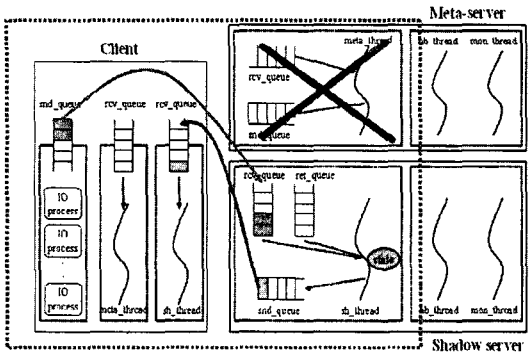
정상 상태에서의 동작 메커니즘은 [그림 3]과 같다. 클라이언트는 meta server와 shadow server에 각각 TCP connection을 맺고 있다. 그래서 클라이언트가 서비스를 받기 위해 request를 각 서버에게 보내게 되는데 request를 두 번 보내게 되는 것이다. Meta server는 클라이언트에서 온 request를 받아 적절히 동작을 수행한 후 그 결과를 클라이언트에게 보낸다. 이때 meta server는 결과를 shadow server에게도 보내게 되며, shadow server는 meta server에서 받은 결과를 가지고 해당 request를 비교해서 shadow server의 state를 변경 시킨다. Shadow server는 meta server와 동일하게 작동을 하며 request를 처리하고 나온 동일한 결과는 클라이언트에게 보내지 않고 자체 내에서 버리게 된다.



[그림 3] 정상 상태에서의 동작 메커니즘

Meta server가 fault가 났을 경우 동작 메커니즘은 [그림

4]와 같다.



[그림 4] Meta server가 fault가 났을 경우 동작 메커니즘

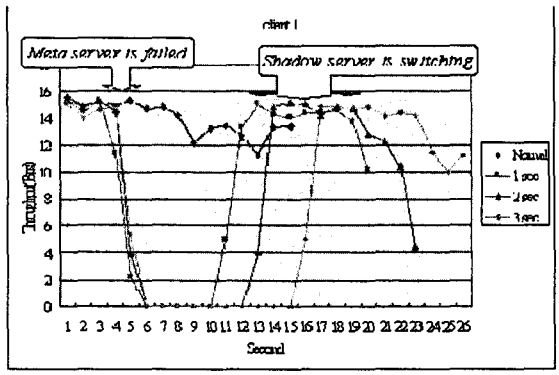
Meta server가 fault가 났을 경우에도 클라이언트와 shadow server는 connection을 미리 맺고 있었으므로 지속적으로 서비스를 해 줄 수 있다. 정상 상태에서의 동작 메커니즘과 차이점은 shadow server에서 나온 결과가 버려지는 것이 아니라 클라이언트에게 보내져 클라이언트가 자신의 일을 계속적으로 수행한다는 것이다.

4. Shadow Server의 실험

Shadow server의 성능을 평가하기 위해 meta server, shadow server 그리고 2대의 클라이언트 총 4대의 컴퓨터를 사용하였다.

meta server가 fault가 났을 경우 shadow server가 meta server의 기능을 제대로 수행하는지를 측정하였다. 실험은 두 대의 클라이언트에서 meta sever로 서비스를 받고 있는 중에 meta server를 꺼 버렸으며, 이 후에 클라이언트가 제대로 서비스를 받는지 측정하였다. 실험을 위해 파일시스템의 성능에 측정에 많이 사용되고 있는 Bonnie benchmark[6]을 수정하여 측정하였다. 클라이언트 1은 200MB를 write하고, 클라이언트 2는 120MB를 write하였다.

[그림 5]는 Meta server가 fault가 났을 경우의 shadow server의 성능을 나타내고 있다. 클라이언트 1에 해당하는 경우만 보여주고 있으며 클라이언트 2도 [그림 5]와 비슷한 유형을 보이고 있다. 각각의 선은 heartbeat의 신호를 받는 주기를 각각 1, 2, 3초로 했을 경우이고, Normal line은 meta server가 fault가 나지 않고 정상적으로 각 클라이언트가 서비스를 받는 경우이다. [그림 5]에서 throughput이 급격히 떨어지는 부분은 meta server가 fault가 난 경우이다. 그리고 일정시간이 지난 후에 throughput이 정상적으로 돌아오는데 이 부분이 shadow server가 meta server의 기능을 대신해 서비스를 해 주는 부분이다. 그런데 shadow server가 서비스를 수행하는 부분이 차이가 있는데 이것은 heartbeat의 주기 때문이다. Heartbeat의 주기가 빠르면 빠를수록 shadow server가 빨리 서비스를 해주는 것을 알 수 있다. 본 논문의 실험에서는 heartbeat의 주기를 1, 2, 3초 하였는데, 1초로 하였을 경우 3.1초에 클라이언트가 서비스를 다시 받음을 실험으로 알 수 있었다.



[그림 5] Meta server가 fault가 났을 경우의 client 1

5. 결론

공유 파일 시스템으로 만들어진 SANfs에서 meta server의 fault는 전체 시스템에 큰 영향을 미친다. 본 논문에서는 meta server의 fault시 생기는 single point-of-failure의 문제를 해결하여 지속적으로 SANfs를 작동시킬 수 있는 fault-tolerant server인 shadow server를 설계 및 구현, 그리고 그 성능을 측정하였다.

참 고 문 헌

[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
[6] Tim Bray. Bonnie. http://www.textuality.com/bonnie/
[7] Matthew T. O'Keefe et al., "Shared File Systems and Fibre Channel" , 1998