ISBN 978-89-6211-310-5

pNFS 기술 동향

일 자: **2008년 12월 5일**
부 서: 슈퍼컴퓨팅사업팀
제출자: 차광호, 조혜영, 김성호

Korea Institute of
Science and Technology Information
www.kisti.re.kr

목 차

1. 개요 1

2. pNFS 3

2.1. 개요 3

2.2. 레이아웃 타입 5

2.3. Layout Operation 6

2.4. 파일 공유 6

2.5. pNFS 클라이언트 7

3. pNFS Prototype 8

3.1. PVFS를 이용한 Prototype 8

3.2. spNFS 8

4. pNFS 활용 연구 11

5. 성능 실험 14

6. 결 론 16

참고문헌 17

1. 개 요

IT 기술의 발달은 다양한 사용자층을 만들어 내면서 다양한 사용자 요구 사항을 만들어 내고 파생되는 데이터양도 날로 급증하고 있다. 이러한 시대적 상황을 반영하여 스토리지 및 파일시스템 역시 다양한 형태로 변화를 꾀하고 있다. 네트워크 파일시스템으로 대표적인 NFS역시 Version 4가 발표되면서 새로운 기능들을 제공하게 되었다. 표 1은 기존 NFSv3와 NFSv4의 특징을 보여준다.

표 1. NFSv3와 NFSv4의 비교

NFSv3	NFSv4
Protocol들의 집합체(mount, lock, status등)	2049 포트를 사용하는 단일 프로토콜
Stateless	Stateful
약한 인증 체계	인증 체계 강화
32bit numeric uid와 gid	스트링으로 기술되는 uid와 gid
UDP, TCP 사용	TCP만 사용
단일 요청 및 단일 응답 체계	복수개의 요청 및 복수의 응답 가능

최근 스토리지 및 파일 시스템 기술에 많은 영향을 끼치고 있는 새로운 요소로서 OSD(Object-based storage device)가 있다. 카네기 멜른 대학에서 개발되었던 NASD(Network Attached Storage Device)의 개념이 발전되어 Panasas사의 ActiveScale Storage Cluster와 SUN의 Lustre와 같은 OSD 파일 시스템으로 발전되었다. OSD의 기본 개념은 파일에 저장되는 데이터와 특성(Attribute)들을 Object로 관리하며 지능화된 저장 장치를 사용한다는 것이다. 즉, 저장 장치가 스스로 데이터를 저장하고 관리할 수 있으므로 기존의 호스트나 서버가 수행하던 저수준의 IO 동작을 저장 장치가 직접 수행할 수 있는 장점이 있다. 특히 NFS와 같은 네트워크

파일시스템에서 IO서버가 관리하는 메타데이터의 약 90%가 Inode 레이어와 같은 저수준 레이아웃에 대한 것이라는 사실을 감안하면 OSD를 통하여 메터데이터 관리의 부하를 상당히 줄일수 있다는 장점이 있다.

이러한 구조적 특징은 클라이언트가 메터데이터 서버를 항시 경유해서 스토리지에 접근하는 기존 방법과 달리 OSD에 직접 접근 할 수 있도록 하는 직접 및 병렬 접근(Direct & Parallel Access)를 가능하게 하였다. 즉, 클라이언트와 스토리지 사이의 데이터 경로(Data Path)와 제어 경로(control Path)를 분리할 수 있도록 하였다.

분산되어 있는 스토리지 자원을 하나의 클러스터로 구성하여 분산 파일 시스템으로 구성하고자 하는 경우 그림 1과 같은 3가지 경우를 고려해 볼 수 있다. Partitioned Data 방법은 관리자가 파일 시스템 명칭공간(name space)를 나누어 각 스토리지에 할당하는 방법으로 각 스토리지 별로 별도의 마운트 포인트를 갖는다. 가장 간단한 방법으로 관리자가 데이터 분배에 관여해야 하는 단점이 있다. Transparent Routing의 경우, 명칭공간(name space)에 대한 일종의 라우팅 기능을 클라이언트와 스토리지 사이에 위치시킨 것으로 관리자의 데이터 분배에 대한 오버헤드를 경감시키는 장점이 있으나 마운트 포인트부분이 병목으로 작용할 수 있어서 확장성에는 제약이 따른다. 이상의 두 방법 모두 대용량 파일을 나누어 저장할 수 없는 관계로 스토리지간의 부하 분산에 한계가 있을 수 있다. Parallel Striped Access는 IO데이터를 스토리지에 나누어 분산시키고 클라이언트가 직접 접근하는 방식으로 병렬 파일 시스템과 같은 HPC용 특수 파일시스템에는 이미 사용되는 기법이나, 일반적인 시스템을 대상으로 한 표준안의 부재가 제약이 된다.

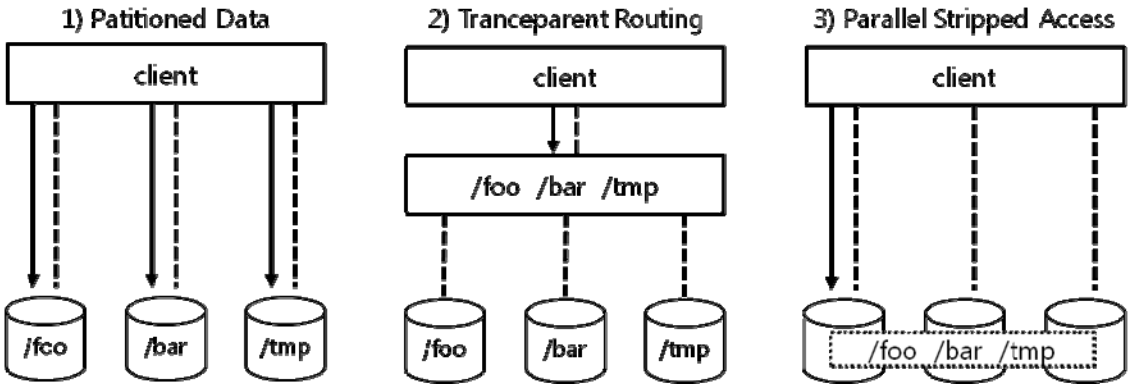

그림1. Clustered File System Services: Three Choices

2. pNFS

2.1. 개요

pNFS는 NFSv4.1의 확장 기능으로 클라이언트가 스토리지에 대한 직접적이고 병렬화된 접근을 허용하여 기존의 NAS시스템이 갖던 확장성 문제를 해결한 표준 프로토콜이다. pNFS는 파일서버, OSD 및 기존의 블록기반 스토리지의 다양한 백엔드 스토리지 시스템을 지원한다. Univ. of Michigan(CITI), EMC, IBM, NetApp, SUN, Panasas 등이 pNFS 커뮤니티에 참여하고 있다. pNFS 의 구성은 다음과 같다.

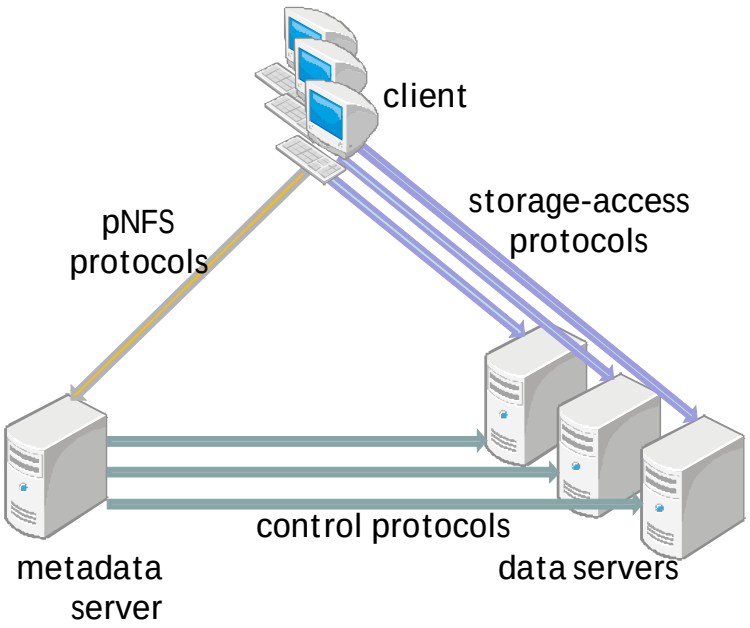

그림 2. pNFS의 구조

pNFS는 클라이언트와 메타데이터 서버간 정보전달에 사용되는 pNFS프로토콜과 클라이언트와 데이터서버(또는 스토리지)간의 데이터 전송에 관여하는 스토리지 접근 프로토콜로 구성된다. pNFS 프로토콜은 NFSv4에 일부 오퍼레이션이 추가된 형태로서 Layout과 데이터 서버의 위치를 질의하고 확인하기 위하여 사용된다. 스토리지 접근 프로토콜은 NFS나 SCSI등이 사용되며 클라이언트가 데이터 서버에 어떻게 접근하는가를 기술한다. 메타 데이터 서버에 존재하는 각 파일에 대한 메타 데이터

는 파일의 위치, 스트라이핑 파라메터, 그 외 토큰등이 포함된 레이아웃과 디렉토리에 대한 파라메터 및 파일 어트리뷰트등을 보유하고 있다. 파일의 삭제등과 같은 오퍼레이션은 클라이언트가 메타데이터 서버를 통하여 수행하게 되며 이때 콘트롤 프로토콜을 통하여 동기화를 수행하게 된다. 현재까지는 분산 파일시스템의 특성을 배려하고 유연성을 고려하여 pNFS 표준에서는 특별하게 제한하고 있지는 않다.

2.2. 레이아웃 타입

A. File Layout

스토리지 서버에 분산 파일시스템이 설치되어 있고, 클라이언트는 각 스토리지 서버의 스트라이프된 파일시스템 데이터를 접근하는 시스템에서 사용된다. 기본적으로는 NFSv4를 스토리지 접근 프로토콜로 권장하고 있으나 일부 프로토 타입의 경우, 스토리지 서버에 설치된 분산 파일시스템을 접근할 수 있는 프로토콜이 사용되기도 한다. 레이아웃 정보에는 데이터 서버 목록, 스트라이프 사이즈, 각 NFS 핸들등이 포함된다. 이러한 File Layout 정보는 간결하고 파일이 갱신되더라도 변경되지 않기 때문에 해당 파일이 광범위하게 공유되더라도 동기화 오버헤드없이 많은 클라이언트에서 캐싱될 수 있다. NetApp., SUN, IBM등이 참여하고 있다.

B. Block Layout

클라이언트가 블록/볼륨 디바이스에 접근하는 기능을 목표로 설계되고 있다. 이때 사용되는 프로토콜로는 iSCSI나 FC(T11 표준)의 사용을 기본으로 한다. 레이아웃 정보에 실제 스토리지 블록이 사용되기 때문에 파일의 갱신시 메타 데이터의 변경이 필요하므로 File Layout과는 달리 메타 데이터 빈번한 요구된다. EMC의 HighRoad가 그 예이다.

C. Object Layout

클라이언트가 OSD에 접근하는 기능을 목표로 설계되고 있다. SCSI Object 프로토콜(T10)을 이용하여 스토리지에 접근하고 레이아웃의 경우 Object ID를 사용하기는 하나, File Layout과 유사한 특징을 갖는다. 현재는 Panasas가 참여하고 있으나 Lustre의 경우에는 아직 미정인 상황이다.

2.3. Layout Operation

앞서 pNFS 프로토콜은 NFSv4에 일부 오퍼레이션이 추가된 형태로서 Layout과 데이터 서버의 위치를 질의하고 확인한다고 하였다. 다음은 대표적인 레이아웃 오퍼레이션의 예이다.

표 2. 대표적인 레이아웃 오퍼레이션

LAYOUTGET	파일 접근을 위한 정보를 구한다.
LAYOUTCOMMIT	레이아웃 정보의 변경을 반영한다.
LAYOUTRETURN	클라이언트가 서버에게 레이아웃 정보를 반환한다.
CB_LAYOUTRECALL	서버가 클라이언트에게 레이아웃 정보의 반환을 요구한다.
GETDEVINFO & GETDEVLIST	초기 마운트시 스토리지 노드에 대한 정보를 요구하고 구한다.

2.4. 파일 공유

파일 공유에 대하여 pNFS는 이전 NFS와의 호환성(Backward compatibility)을 염두에 두어 close-to-open consistency를 제공한다. 즉 변경된 파일이 close되는 시점에서나 다른 클라이언트들에게 변경된 내용이 나타나게 되는 방식으로 쓰기-공유(Write-Sharing)이 흔하지 않다는 가정을 전제로 하고 있다. 쓰기-공유(Write-Sharing)이 나타나는 경우에는 NFSv4가 사용하는 방식인 메타 데이터 서버를 이용하여 바이트 단위 잠금(byte-range lock)을 수행하게 되므로 쓰기의 대한 직렬화가 나타나게 된다.

2.5. pNFS 클라이언트

다음 그림은 pNFS 클라이언트의 구조를 보여 주고 있다. NFSv4.1 하부에 레이아웃 캐싱과 같은 일반적인 레이아웃 드라이버가 위치하고 레이아웃 타입별로 세분화된 레이아웃 드라이버가 위치한다.

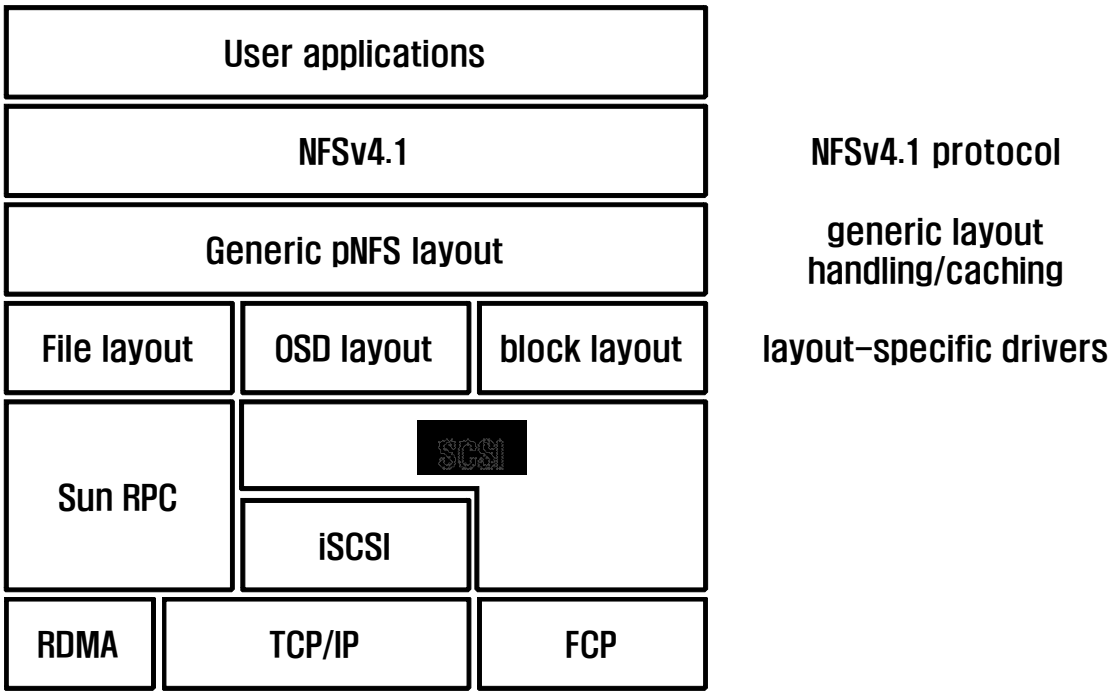

그림 3. pNFS 클라이언트의 구조

3. pNFS Prototype

3.1. PVFS를 이용한 Prototype

다음 그림은 초기 미시간대학교(Univ. of Michigan)에서 개발한 프로토타입으로 파일 레이아웃을 지원하는 형태로서 PVFS를 이용한 형태로 스토리지 노드에 PVFS를 설치하고 pNFS 클라이언트는 PVFS 레이아웃 드라이버를 이용하여 스토리지에 접근할 수 있도록 하였다. pNFS의 최초 모델이며 업그레이드가 중단된 상태이다.

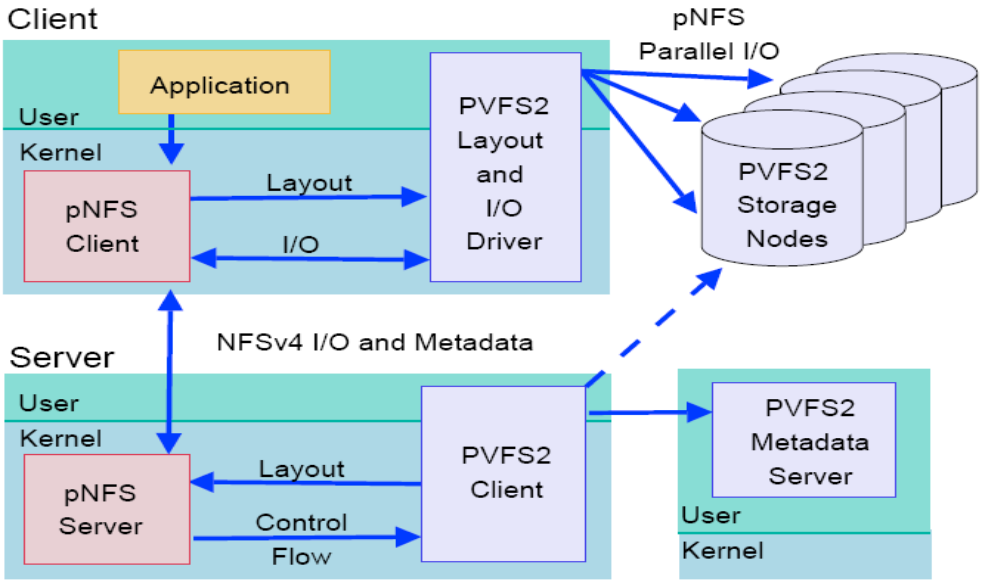

그림 4. PVFS기반 pNFS Prototype

3.2. spNFS

PVFS를 이용한 프로토타입의 복잡한 구조를 탈피하고 단순함을 강조한 형태의 pNFS 솔루션이 spNFS(Simple pNFS)이다. NetApp.에서 개발하고 있으며 간단한 형태의 파일시스템 연산을 제공하면서 pNFS를 보다 쉽게 구성하는데 중점을 두고 있다. 다음 그림5 는 spNFS의 구성 예이다.

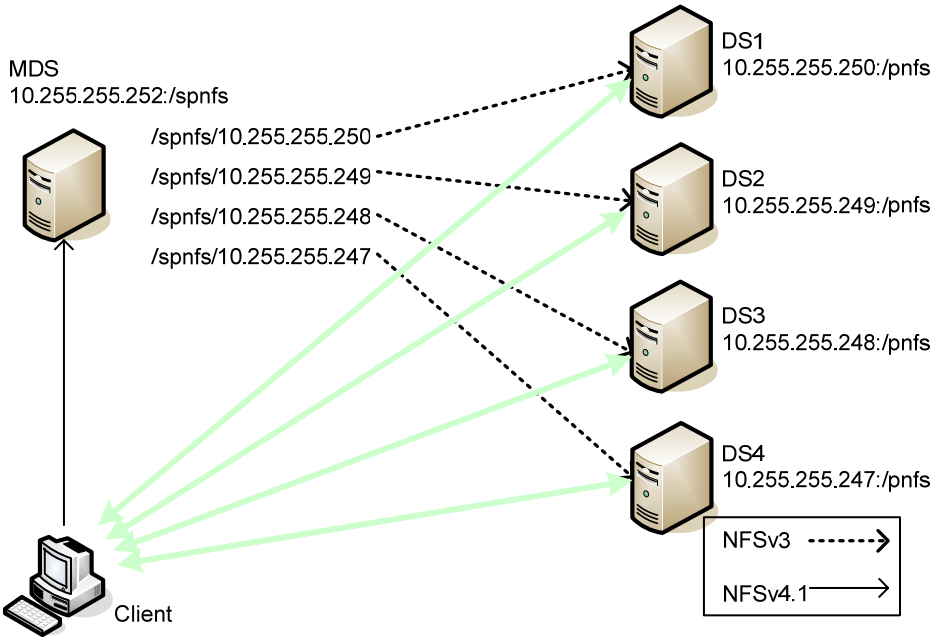

그림 5. spNFS의 구성예

spNFS는 별도의 파일 시스템을 필요로 하지 않는다는 특징이 있는데 기존의 NFSv3프로토콜을 스토리지 접근 프로토콜로 이용하여 각 데이터 서버에 접근하게 된다. 이때 MDS 서버에 이러한 구성을 가능케 하는 spnfsd가 동작하고 있어야 하며 'spnfsd.conf'설정 파일을 이용하여 시스템의 구성을 기술하게 된다.

```
[General]

Verbosity = 1
Stripe-size = 8192
Dense-striping = 0
Pipefs-Directory = /var/lib/nfs/rpc_pipefs
DS-Mount-Directory = /spnfs
```

```
[DataServers]

NumDS = 4

DS1_IP = 10.255.255.250
DS1_PORT = 2049
DS1_ROOT = /pnfs
DS1_ID = 1

DS2_IP = 10.255.255.249
DS2_PORT = 2049
DS2_ROOT = /pnfs
DS2_ID = 2

DS3_IP = 10.255.255.248
DS3_PORT = 2049
DS3_ROOT = /pnfs
DS3_ID = 3

DS4_IP = 10.255.255.247
DS4_PORT = 2049
DS4_ROOT = /pnfs
DS4_ID = 4
```

그림 6. spnfsd.conf

4. pNFS 활용 연구

4.1. Large Files, Small Writes, and pNFS

과학 기술용 IO작업이 대규모 파일을 대상으로도 하지만 빈번한 소규모 쓰기 연산이 있다는 점을 감안하여 pNFS 프로토 타입을 개선한 연구이다. 즉 Write 요청의 threshold값을 기준으로 소규모 파일에 대한 연산은 pNFS메타 데이터 서버가 직접 수행하고, 그 이외의 경우에는, 클라이언트가 직접 IO 작업을 수행하도록 하였다.

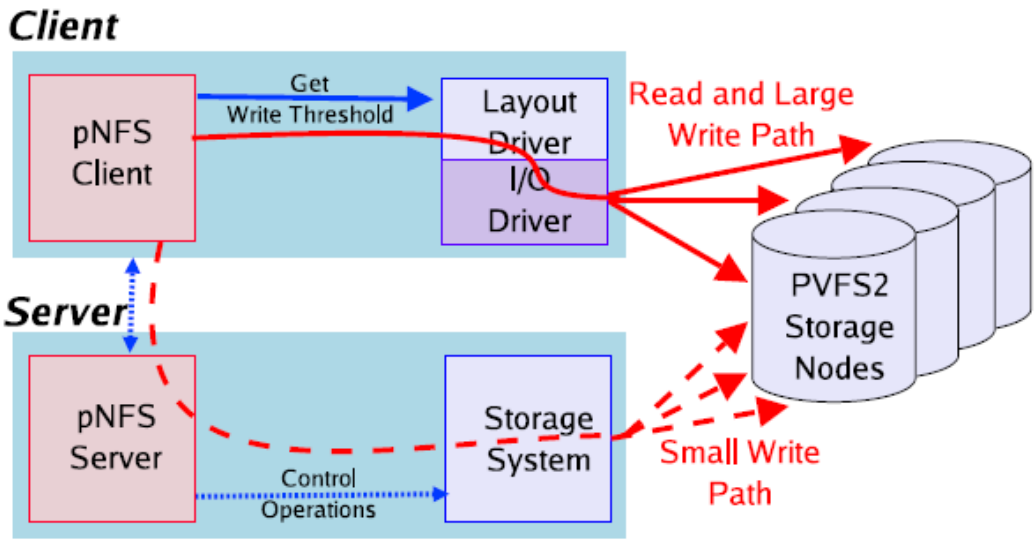

그림 7. 쓰기 임계값에 따른 접근 경로 차별화

처리해야 하는 데이터양에 따라 쓰기 경로를 다르게 하여 특정 구간에서의 성능 저하를 최소화하고자 연구로서 실제 실험 결과에서는 데이터 크기에 따라 pNFS와 PVSF의 장점을 조합적으로 취한 성능을 보여주었다.

4.2. WAN 환경에서의 pNFS

pNFS의 적용 분야를 확대시키는 다양한 연구가 있었는데 위에 설명한 대용량 IO 이외에 소규모 IO에서도 성능 저하가 나타나지 않도록 경계 값을 이용하는 하이브리드 형태의 pNFS가 그 예이다. 또한 최근에는 WAN을 사이에 두고 원격지의 파일 시스템에 접근하는 수단으로 pNFS를 사용한 사례가 보고되고 있어 향후, 사용 분야가 더욱 확대될 것으로 예상된다.

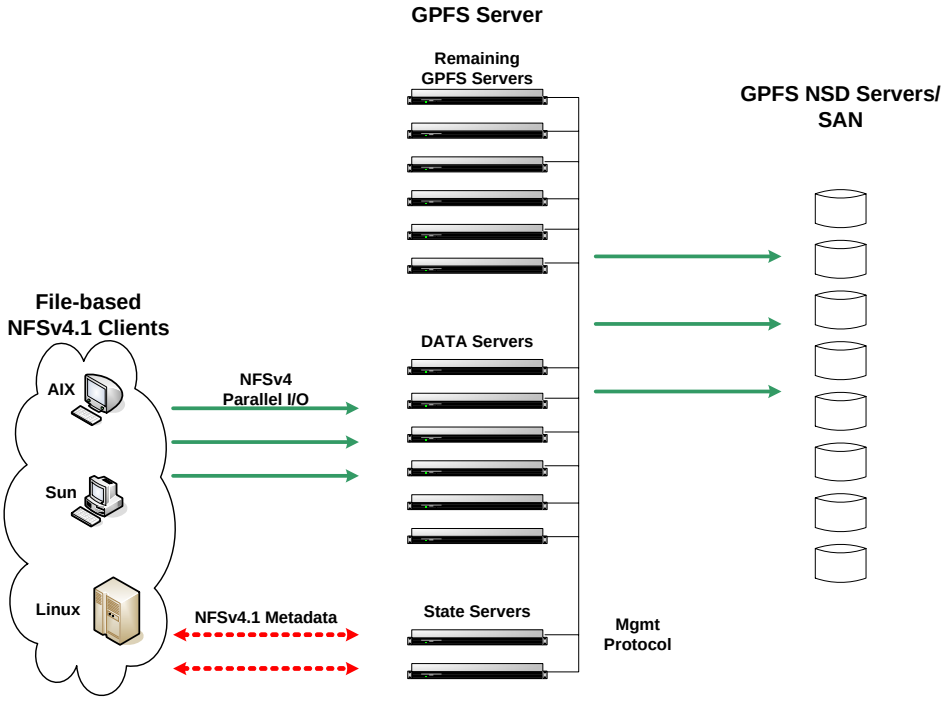

그림 8. pNFS를 이용한 WAN환경에서의 GPFS 접근예

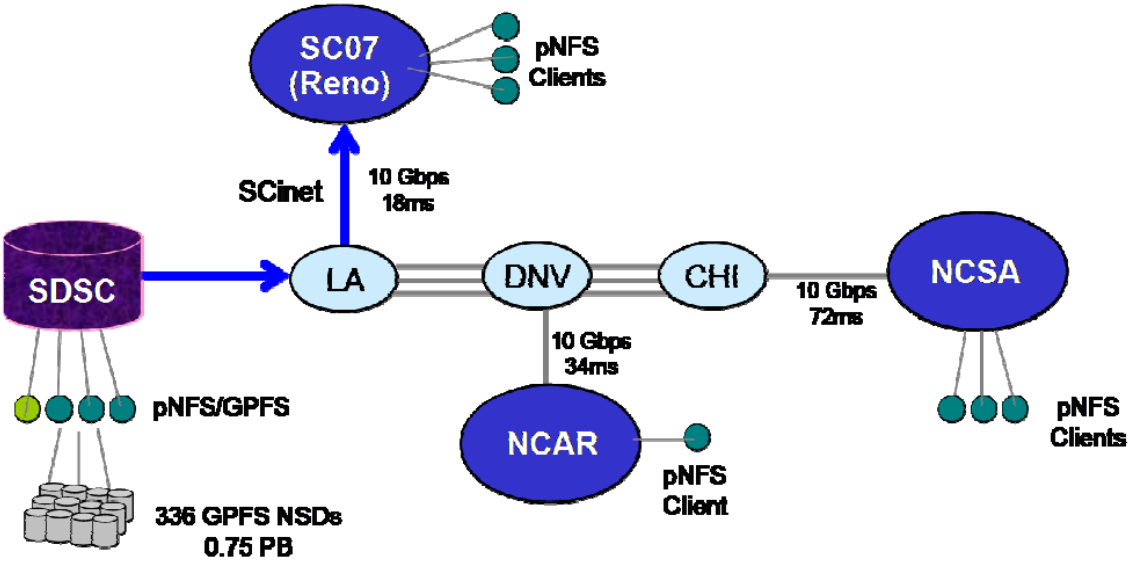

그림 9. pNFS를 TeraGrid에 이용한 사례

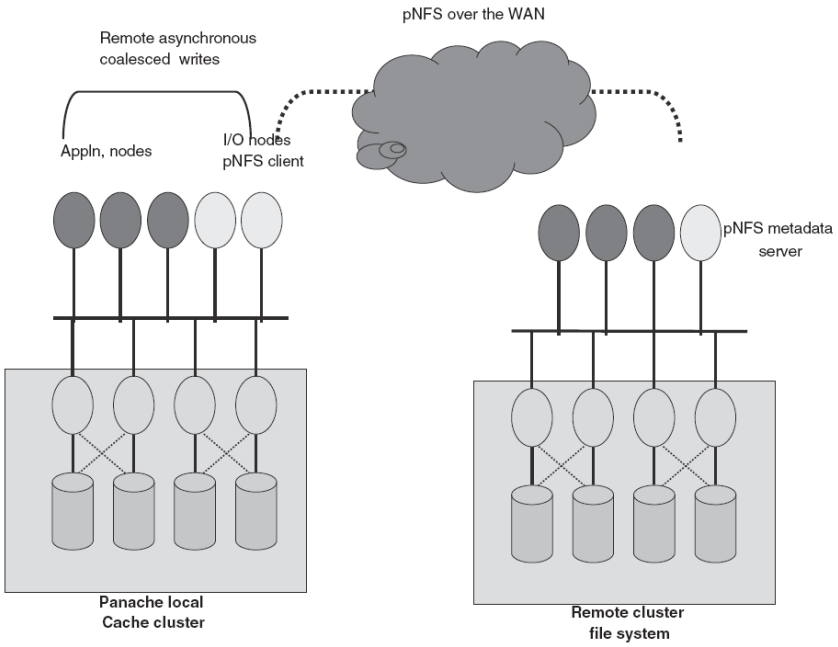

그림 10. pNFS의 WAN에서 사용 예

그림10는 WAN상의 원격지 파일 시스템을 로컬에 위치한 파일 시스템에 캐싱하기 위한 Panache 아키텍쳐로서 원격지 파일 시스템에 접근하는 과정에서 pNFS를 이용하여 데이터를 전달하는 구조를 취하고 있다.

5. 성능 실험

본 장에서는 대표적인 IO 벤치마크 도구인 Bonnie를 이용하여 spNFS에서 수행한 성능 측정 결과를 설명한다. 표 3은 spNFS의 테스트 환경을 구축하는데 사용된 소규모 클러스터 시스템의 구성을 보여 주고 있으며 실제 환경 설정은 그림 5와 같다.

표 3. 테스트 베드 구성 요소

1 Client, 1 MDS, and 4 Data Servers	
CPU	Intel Pentium 4 XEON 2.8
Memory	1GB
HDD	80GB SATA
Networks	Fast Ethernet, Gigabit Ethernet
OS	Linux - 2.6.25 spnfs patch

5.1. Stripe 크기 변경에 따른 성능 변화

분산 파일 시스템이나 RAID의 경우에 분할 및 저장 단위인 stripe의 크기가 전체 성능에 영향을 주게 된다. 본 장에서는 stripe 크기의 변화에 따른 성능 변화를 살펴 보았다.

spNFS를 사용하기 위해서는 파일 시스템 관련 데몬인 spnfsd가 메타 데이터 서버에서 실행되어야 하고 이때 사용되는 spnfsd용 환경 설정 파일을 수정하여 스트라이핑 크기(stripe-size)를 변경할 수 있다.

그림 11는 스트라이핑 크기를 4KB에서 32KB까지 증가시키면서 bonnie 테스트의 블록 쓰기와 블록 읽기를 수행한 결과이다. 스트라이핑 크기가 증가할수록 향상된 결과를 보여 주었으나 8KB일 때 개선 폭이 가장 크다는 것을 알 수 있다. 즉 8KB 이상의 스트라이핑 크기에서는 개선 폭이 크지 않아 이미 포화 구간에 들어선 것으로 볼 수 있다.

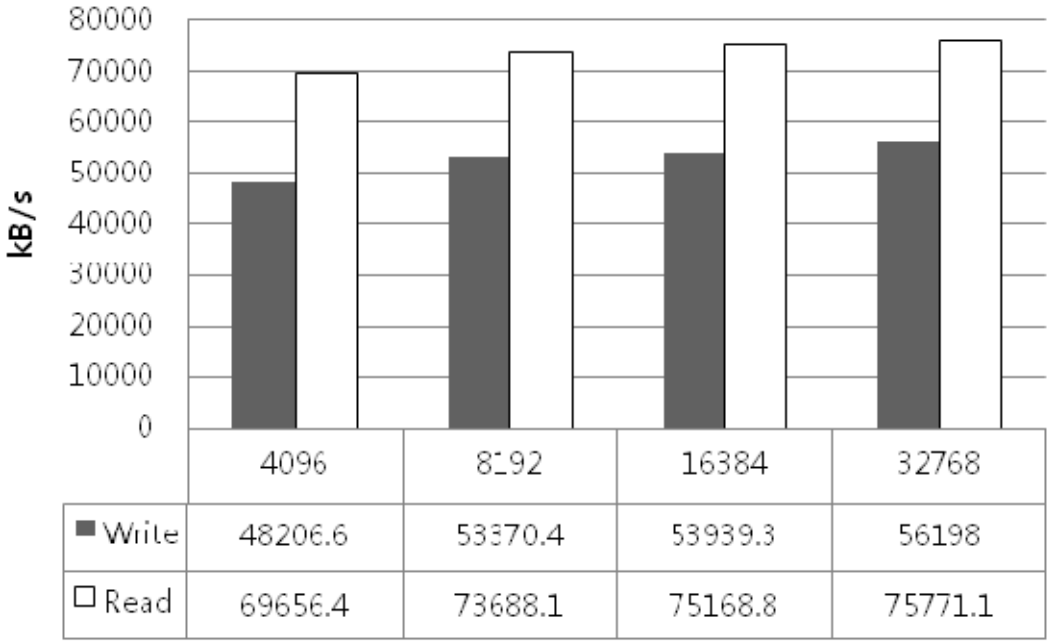

그림 11. Stripe-size별 성능 변화

6. 결 론

병렬화된 스토리지에 대한 직접 접근은 IO 데이터를 스토리지에 나누어 분산시키고 클라이언트가 직접 접근하는 방식으로 병렬 파일 시스템과 같은 HPC 용 특수 파일 시스템에는 이미 사용되는 기법이나, 일반적인 시스템을 대상으로 한 표준안의 부재가 제약이 된다. pNFS(Parallel NFS)는 이러한 문제를 해결하기 위하여 제시되는 새로운 NFS 기술이다. 본 보고서에서는 pNFS 의 연구 동향과 더불어 소규모 클러스터 시스템에서의 나타내는 성능적 특징을 조사하였다.

스트라이핑 크기를 4KB 에서 32KB 까지 증가시키면서 성능을 조사한 결과, 스트라이핑 크기가 증가할수록 향상된 결과를 보여 주었으나 8KB 일 때 가장 효율적인 성능을 보여 주었다.

참 고 문 헌

[1] Brian Pawlowski, Spencer Shepler, Carl Beame, Brent Callaghan, Michael Eisler, David Noveck, David Robinson, Robert Thurlow, "The NFS Version 4 Protocol," Proc. of the 2nd International System Administration and Networking Conference, 2000, http://www.nluug.nl/events/sane2000/papers/pawlowski.pdf

[2] Garth Goodson, Sai Susharla, Rahul Iyer, "Standardizing storage clusters," ACM Queue, Vol.5(6), pp. 20 ~ 27, 2007

[3] "NFSv4 Wiki," Retrieved Aug. 28, 2008, from http://wiki.linux-nfs.org/wiki/index.php/Main_Page

[4] Dan Muntz, Mike Sager, Ricardo Labiaga, "A Simple pNFS Server," Retrieved Aug. 28, 2008, from http://www.connectathon.org/talks08/dmuntz-spnfs-cthon08.pdf

[5] David Nagle, Denis Serenyi, Abbie Matthews, "The Panasas ActiveScale Storage Cluster - Delivering Scalable High Bandwidth Storage," Proc. of the 2004 ACM/IEEE conference on Supercomputing, pp. 53, 2004.

[6] Dean Hildebrand, Lee Ward, Peter Honeyman, "Large files, small writes, and pNFS," Proc. of the 20th ACM International Conference on Supercomputing, pp. 116 ~ 124, 2006.

[7] Dean Hildebrand, Peter Honeyman, "Direct-pNFS: scalable, transparent, and versatile access to parallel file systems," Proc. of the 16th ACM International Symposium on High performance Distributed Computing, pp.

199 ~ 208, 2007

[8] Dean Hildebrand, Peter Honeyman, "Exporting Storage Systems in a Scalable Manner with pNFS," Proc. of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies, pp. 18 ~ 27, 2005.

[9] Dean Hildebrand, Marc Eshel, Roger Haskin, Phil Andrews, Patricia Kovatch, John White, "Deploying pNFS Across the WAN: First Steps in HPC Grid Computing," The 9th LCI International Conference on High-Performance Clustered Computing, http://www.linuxclustersinstitute.org/conferences/archive/2008/PDF/Hildebrand_98265.pdf

[10] R. Ananthanarayanan, M. Eshel, R. Haskin, M. Naik, F. Schmuck, R. Tewari, "Panache: a parallel WAN cache for clustered filesystems," ACM SIGOPS Operating Systems Review, Vol.42(1), pp. 48 ~ 53, 2008.

[11] David L. Black, Stephen Fridella & jason Glasgow (EMC). pNFS Block/Volume Layout (draft 09), June 11, 2008. http://tools.ietf.org/pdf/draft-ietf-nfsv4-pnfs-block-09.pdf.

[12] B. Halevy, B. Welch, J. Zelenka, Panasas, Object-based pNFS Operations (draft 09), June 11, 2008. http://tools.ietf.org/pdf/draft-ietf-nfsv4-pnfs-obj-09.pdf