ETRI

전자통신동향분석 제26권 제5호 2011년 10월

pNFS 표준화 및 연구개발 동향

Current Status and Outlook of pNFS and Its Implementation

소프트웨어 기술의 미래전망 특집

박정숙 (J.S. Park) 저장시스템연구팀 선임연구원
김수영 (S.Y. Kim) 저장시스템연구팀 선임연구원
차명훈 (M.H. Cha) 저장시스템연구팀 선임연구원
김동오 (D.O. Kim) 저장시스템연구팀 선임연구원
김영창 (Y.C. Kim) 저장시스템연구팀 선임연구원
김홍연 (H.Y. Kim) 저장시스템연구팀 책임연구원

목 차

요즘 신산업으로 떠오르고 있는 개인 유전체 분석이나 차세대 시퀀싱 기술과 같은 고성능 컴퓨팅 응용들은 data-intensive한 작업들을 요구하며, 이러한 응용을 지원하기 위한 고성능 파일 시스템 기술에 대한 연구들이 다수 진행 중이다. 그러나 그 결과물들은 업체별로 보유한 고유 기술로서, 상호 호환성 등의 문제로 인해 표준화의 필요성이 제기되어 왔다. 현재 파일 시스템과 관련하여 거의 모든 IT 업체에서 사용하고 있는 실질적인 표준은 NFS(Network File System)이다. IETF(Internet Engineering Task Force)에서는 이러한 요구사항들을 반영하여 2010년에 NFSv4.1 표준을 공표하였고, 특히 I/O 성능을 향상시키기 위한 pNFS(parallel NFS)는 NFSv4.1의 핵심 기능으로서 다수 업체들과 연구기관들에서 많은 관심을 받고 있다. 본 고에서는 pNFS 표준화 및 연구개발 동향과 관련된 이슈들에 대해 기술하고자 한다.

Ⅰ. 서론

IT 관련 기술의 진보는 이제 개인 유전체 분석이나 차세대 시퀀싱 기술과 같은 바이오 기술, 항공 우주 기술 등에 폭넓게 적용되고 있다. 이러한 응용들은 슈퍼 컴퓨터 및 컴퓨터 클러스터링과 같은 고성능 컴퓨팅 기술(HPC: High Performance Computing) 및 고성능 입출력 기술을 필요로 하며, 특히 data-intensive한 작업들을 많이 필요로 한다. 이에 따라 Lustre, GPFS(General Parallel File System)와 같은 병렬 파일 시스템 기술에 대한 연구들이 활발하게 진행되고 있다. 그러나 이러한 기술들은 업체별 고유 기술들로서, 상호 호환성, 확장성 등의 문제로 인해 표준화의 필요성이 강하게 제기되어 왔다[1],[2],[3].

지금까지 분산 파일 시스템과 관련해서 업체들이 실제로 사용하고 있는 표준은 NFS(Network File System)가 유일하다. 1980년대에 하드디스크가 없는 환경에서의 작업을 지원하기 위해 Sun사에서 개발을 시작한 NFS 기술은 현재는 원격 파일 시스템의 표준으로서 스토리지가 필요한 거의 모든 업체에서 사용하고 있고, 지금까지 NFSv3는 안정적인 서비스를 제공하여 왔다.

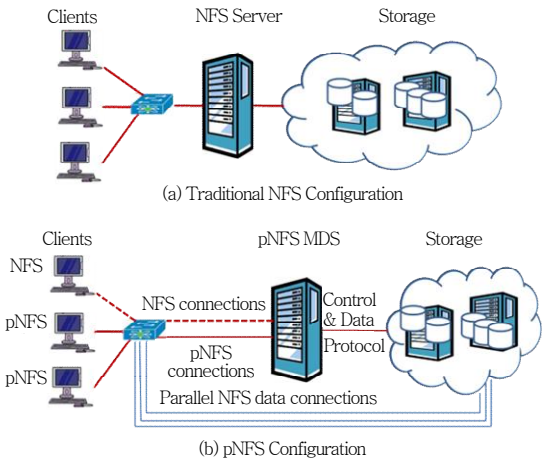
그런데, 2000년대 들어오면서 인터넷 사용량이 급격이 증가되고 사용 패턴들도 다양해짐에 따라서 NFS의 성능 부족 문제가 대두되기 시작했다. 이를 해결하기 위해 NFSv3 기반의 클러스터링 파일 서버들을 포함하여 다양한 개선 노력이 있었으나, 성능과 확장성 측면에서 한계점에 다다르게 되었다. 현재 스토리지 업체들은 다양한 제품을 보유하고 있지만, 이러한 제품들은 상호호환성 문제를 가지고 있다. 오픈소스인 GFS(Global File System)나 Lustre 등도 마찬가지다[4].

이러한 문제점을 해결하기 위하여 IETF(Internet Engineering Task Force)의 NFSv4 WG에서 표준화 작업을 진행해 오고 있다. 확장성, 성능, 시큐리티 기능 강화, 다양한 레이아웃 지원 등과 같은 HPC와 관련한 문제에 대한 대안들을 제시한 NFS의 새로운 표준인 NFSv4.1은 2010년 RFC5661 표준으로 공표되었다. NFSv4.1의 특성 중에서도 pNFS(parallel NFS) 기술은 데이터를 다수의 저장 서버들에 분산 저장하고 액세스를 병렬로 수행하게 함으로써 획기적인 성능 개선을 가능하게 한 기술이다[5],[6],[7].

(그림 1)과 같이, 기존의 NFS 환경에서는 클라이언트 수가 증가할수록 단일 서버로 인한 I/O 병목 현상이 심각하였으나, pNFS 환경에서는 다수의 DS(Data Server)들이 늘어나는 클라이언트들의 요청을 병렬로 처리할 수 있는 구조를 취함으로써 이러한 문제를 해결하였다[1].

pNFS를 포함한 NFSv4.1 표준에 대해서는 아직까지 구현이 진행 중에 있으므로, 본 고에서는 표준의 세부 내용을 중심으로 이슈들을 소개하고 구현 현황을 소개하고자 한다.

본 고의 목차는 다음과 같다. 제Ⅱ장에서는 NFSv4.1의 표준화 진행 현황을 소개한다. 제Ⅲ장에서는 NFSv4.1 중에서도 업체들의 주요 관심사가 되고 있



(그림 1) NFS와 pNFS의 구조 비교

는 pNFS에 대해 소개한다. 제Ⅳ장에서는 pNFS의 기술 개발 현황 및 관련 이슈들을 소개한다. 마지막으로 제Ⅴ장에서는 pNFS의 향후 방향에 대해 논하고자 한다.

Ⅱ. 표준화 진행 현황

pNFS를 포함한 NFSv4.1에 대한 표준화는 IETF의 NFSv4 워킹그룹에서 진행 중인데[8], 본 장에서는 NFSv4.1의 표준화 현황 및 표준 내용을 기술하고자 한다.

1. IETF NFSv4 워킹그룹

1995년에 RFC1813(NFSv3) 표준이 공표된 후, NFSv3는 원격 파일 시스템의 표준으로 업체들에 의해 널리 사용되었다. 그 후 인터넷 환경의 변화 등 다양한 요구에 의해 2003년 4월에 RFC3530(NFSv4) 표준이 공표되었다. NFSv4에는 시큐리티 기능 강화, 방화벽 고려, 캐시 메커니즘의 강화, 비UNIX 지원, 시스템 고장 시 복구 기능, 관리 용이성, 권한 위임(delegation) 등의 특성들이 추가되었다[9],[10],[11].

그러나 보다 큰 변화는 RFC5661로 명명되는 NFSv4.1에 의해 발생되었는데, 그 중 가장 핵심은 NFS에 병렬 I/O를 지원하여 성능을 향상시키고자 한 pNFS이다[7].

실제로 업체들은 일찍부터 이 문제에 대해 고민하기 시작했는데, 2003년에 Panasas, NetApp, IBM, EMC, SUN, CITI[12]를 포함한 주요 업체들 간에 pNFS에 관한 첫 회의가 이루어졌다. 2005년에는 pNFS 내용이 NFSv4.1 드래프트에 포함되었고, 2008년에 NFSv4.1 드래프트가 표준 트랙에 포함되었다. 그 후 여러 번의 검증 작업을 통해 2010년 1월에 NFSv4.1은 RFC5661로 등록되었다[8],[13]. CITI, EMC, NetApp, Panasas가 클라이언트 및 서버 표준화 작업에 참여하였고, IBM은 클라이언트 표준화 작업에 참여하였다. <표 1>은 pNFS 표준화를 위한 이러한 노력들이 어떤 과정을 거쳐왔는지 보여준다[13].

NFSv4.1과 관련한 표준들은 <표 2>와 같이 NFSv4.1 특성 및 동작을 기술한 표준인 RFC5661, XDR(External Data Representation)을 기술한 RFC 5662, 블록 레이아웃에 대해 기술한 RFC5663 및 객체 레이아웃에 대해 기술한 RFC5664를 포함하여 총 4개로 구성된다[8],[14], [15],[16],[17].

pNFS 환경에서는 파일, 블록, 객체를 포함하여 세

〈표 1〉 pNFS 표준화 진행 경과

Data	Activities
2003	RFC publication of NFSv4.0(RFC3530, RFC 3531)
2004	CMU, NetApp and Panasas draft pNFS Problem and Requirement statements
2005	CITI, EMC, NetApp and Panasas draft pNFS extensions to NFS
2005	NetApp and Sun Demonstrate pNFS at Connectathon
2005	pNFS added to NFSv4.1 draft
2006~2008	Specification baked(26 iterations of NFSv4.1/ pNFS spec)
2008	NFSv.4.1/pNFS reaches IETF Last Call
2010	RFC publication of NFSv.4.1 related documents(RFC5661-RFC5667)

〈표 2〉 NFSv4.1 관련 표준

RFC no	Standard Name	Layout
RFC 5661	Network File System(NFS) Version 4 Minor Version 1 Protocol	File
RFC 5662	Network Flie System(NFS) Version 4 Minor Version 1 External Data Representation(XDR) Description	-
RFC 5663	Parallel NFS(pNFS) Block/Volume Layout	Block
RFC 5664	Object-Based Parallel NFS(pNFS) Operations	Object

가지 데이터 유형을 사용할 수 있는데, RFC5661은 그 중 파일 데이터 유형을 기반으로 pNFS를 포함하여 NFSv4.1의 프레임워크 및 모든 오퍼레이션들에 대해 기술하고 있다.

그 외에, 성능 향상을 위해서 IB(InfiniBand) 상에서 RDMA(Remote Data Memory Access)를 지원하는 이슈와 관련하여 RFC5666, RFC5667 표준이 NFSv4.1과 함께 2010년에 공표되었다[8].

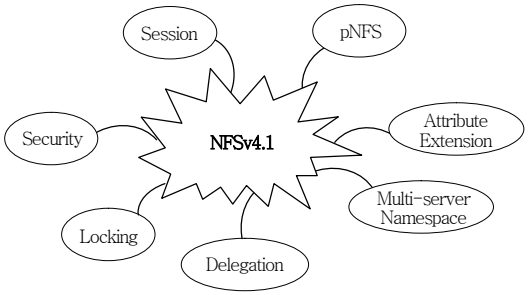
2. NFSv4.1의 특성

NFSv3에서는 클라이언트가 서버에 접속하여 서비스를 받기 위해서는 mountd를 통해 파일 핸들을 가져오고, 파일 핸들을 사용하여 portmapper에 접속하여 NFS 서버의 포트 번호를 가져온 후 nfsd에 접속하는 일련의 과정을 거친다.

그러나 NFSv4.1에서는 이러한 프로토콜들이 모두 nfsd 하나로 통합되었다. 또한 NFSv4.1은 LAN 또는 인터넷 상에서도 동작하도록 설계되었다. 인터넷은 보안이 안되고 느리며 이기종 네트워크 환경이므로, 이러한 문제점들을 보완하기 위한 특성들이 추가되어 NFSv3에 비해 훨씬 견고해졌고 LAN 상에서도 NFSv3보다 잘 동작한다[10].

〈표 3〉 NFSv3와 NFSv4.1의 특성 비교

NFSv3	NFSv4.1
• Security: AUTH_SYS	• Security: Kerberos v5, SPKM, LIPKEY
• Additional protocol: NLM, NSM • Portmapper: allocating a new port to nfsd for client's connection • UDP • Stateless • UNIX supporting	• NO additional protocol: integrated into nfsd. Only rpc.nfsd & idmapd • Fixed port: 2049(firewall friendly) • Pseudo file system & Virtual root directorly(fsid): providing a unique access to exported data • TCP(reliability supporting) • UID/GUID with named string types • Crash recovery • Stateful(concept of sessions) • Non-UNIX supporting: UNIX, Windows
	• pNFC: Parallel access to FS
	• Easy to administer: replication, migration, referral
	• Delegation: file delegation(Read, Write), directory delegation(read)



(그림 2) NFSv4.1의 주요 특성들

NFSv4.1이 기존 버전인 NFSv3에 비해 어떤 다른 특성을 가지는지는 <표 3>을 통해 확인할 수 있다[11],[12].

NFSv4.1에서 지원하는 주요 특성들은 (그림 2)와 같다. 각 특성들과 관련한 핵심 사항들을 요약하면 다음과 같다[14].

가. Session

분산 파일 시스템에서 연결 손실, 클라이언트나 서버의 장애 복구 등에 대응하기 위해 NFSv4.1에서 추가된 주요한 특성이다. NFSv4.1에서의 오퍼레이션들은 세션을 만드는 경우를 제외하고는 모두 세션 기반에서 동작한다.

나. pNFS

pNFS 특성은 병렬 액세스를 통한 성능 개선을 달성하기 위해 NFSv4.1에 추가된 기능이다. 파일, 블록, 객체 등의 데이터 유형을 지원할 수 있다.

다. Attribute Extension

NFSv4.1은 사용자의 필요에 따라 어트리뷰트들을 마음대로 추가할 수 있도록 유연한 구조를 제공한다.

라. Multi-Server Namespace

NFSv4.1에서는 파일 시스템의 관리를 위해 migration, replication, referral의 메커니즘을 제공한다[18]. 이 특성은 특정한 서버에 접근했을 때 원하는 파일들이 없거나 그 서버에 장애가 발생하는 경우 다른 서버로 찾아가서 원하는 파일을 찾아낼 수 있도록 지원하기 위한 목적을 가진다.

마. Delegation

권한 위임(delegation)은 서버(MDS: MetaData Server)에서 이루어져야 하는 의사 결정을 클라이언트에게 위임하는 메커니즘을 의미한다. 위임은 클라이언트 캐싱 기능을 제공함으로써 클라이언트와 서버 사이의 데이터 송수신량을 줄이고 시간을 절약하게 함으로써 성능 개선에 큰 이득이 된다. NFSv4.1에서는 파일 권한 위임(file delegation)과 디렉토리 권한 위임(directory delegation)을 제공하는데, 파일의 경우에는 Read/Write 모드에 대해서 위임 기능을 제공하고, 디렉토리의 경우에는 Read 모드에 대해서만 위임 기능을 사용할 수 있다.

바. Locking

NFSv4.1에서 locking 기능은 NFSv3과는 달리 nfsd 서버 코드에 통합하여 제공한다. locking 기능은 byte-range lock, share reservation, file delegation, directory delegation, layout을 포함하여 5종류의 lock을 지원한다. 단, byte-range lock은 별도의 LOCK 오퍼레이션에 의해 성립되지만, 그 외의 나머지는 파일을 여는 OPEN 오퍼레이션에 의해 설정될 수 있다.

사. Security

NFSv4.1은 보안 기능을 제공하기 위해 RPCSEC-GSS(Generic Security Service) 프레임워크를 도입하고, Kerberos v5, SPKM(Simple Public-key GSS-API Mechanism), LIBKEY 등을 사용한다.

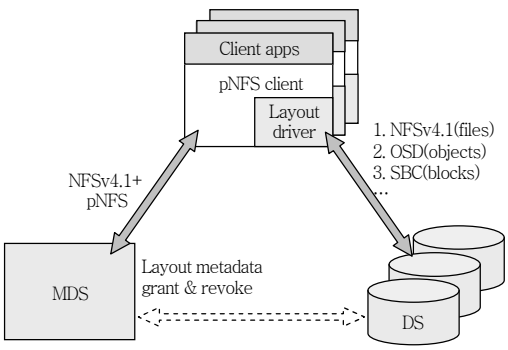
Ⅲ. pNFS

pNFS는 NFSv4.1에 포함되어 있는 핵심 특성으로서, 데이터 I/O를 병렬화함으로써 성능을 극대화하기 위한 프레임워크 및 기능을 정의한다. 본 장에서는 RFC5661에 기술된 pNFS 내용을 소개하고자 한다[14].

1. pNFS 구조

pNFS는 (그림 3)과 같이, 기본적으로 단일의 메타데이터 서버(MDS), 다수 개의 데이터 서버(DS), 다수 개의 클라이언트들로 구성된다[13],[14].

실제 데이터는 DS들에 분산 저장하고, 이들에 관한 메타데이터들을 MDS에 저장한다. 메타데이터는 파일의 실제 저장소와 관련한 정보를 가지고 있다. 클라이언트가 데이터에 접근하기 위해서는 MDS로부터 관련 레이아웃을 얻어와야 한다. 클라이언트는 MDS로부터 얻어온 레이아웃의 정보를 기반으로 DS와 직접 통신하여 데이터 I/O를 수행하는데, DS와의 통신은 병렬로 수행된다. 이런 방식으로 data path와 control path를 분리함으로써 pNFS는 분산 파일 시



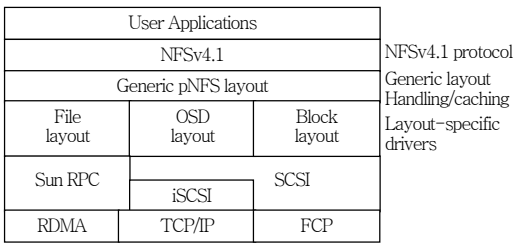
(그림 3) pNFS 기본 구조

스템의 성능을 향상할 수 있다.

클라이언트와 MDS와의 통신을 위해 pNFS 프로토콜이 제공된다. 또한 클라이언트와 DS의 통신을 위해서는 데이터 유형에 맞는 프로토콜을 사용할 수 있는데, 이를 위해서 레이아웃(layout) 개념이 제공된다. 레이아웃은 데이터 유형에 따라 정의될 수 있는데, 전술한 바와 같이 표준에서는 파일, 블록, 객체의 세 종류를 지원하고 있다.

RFC5661에서는 1개의 MDS를 고려하고 있으며, 다수의 MDS로 확장하는 부분은 특별히 언급하고 있지 않다[5]. 또한 MDS와 DS 사이의 프로토콜은 업체별 특성을 살릴 수 있도록 표준에 포함하고 있지 않다.

pNFS 클라이언트에서의 프로토콜 스택 구조는 (그림 4)와 같다[19]. 그림에서와 같이 클라이언트 코드는 세가지 레이아웃을 모두 지원하고, 서버가 지원하는 데이터 유형에 맞추어 레이아웃을 사용할 수 있다.



(그림 4) pNFS 클라이언트 프로토콜 스택

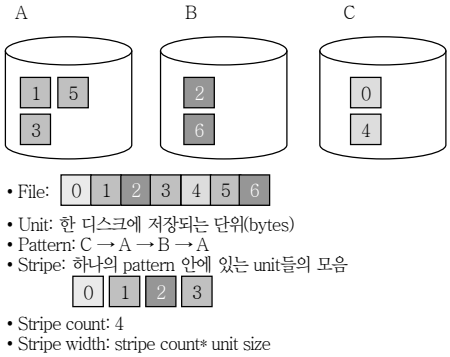
2. 레이아웃

레이아웃(layout)은 한 파일의 데이터가 다수의 스토리지에 어떤 방식으로 저장되는가를 정의하고 있는 구조체로서, MDS가 관리한다[14]. 클라이언트가 파일에 I/O를 하고 싶은 경우에는 먼저 MDS에 레이아웃을 요청해야 한다. 따라서 레이아웃의 구조를 상세히 이해함으로써 pNFS의 동작을 쉽게 이해할 수 있다. 본 절에서는 파일 레이아웃에 대해 설명한다.

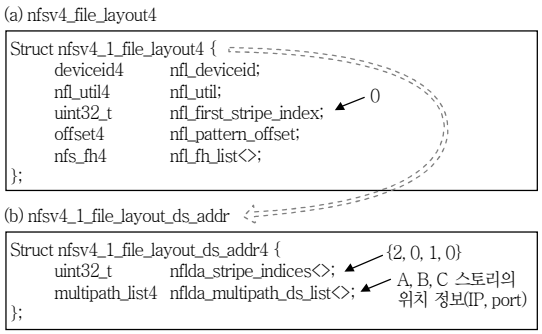
pNFS에서의 레이아웃의 개념은 (그림 5)에 나타나 있다. 한 저장소에 저장되는 최소 단위를 unit이라 하고, 하나 이상의 저장소에 어떤 식으로 배치시킬 것인가 정하는 것을 pattern이라 한다. (그림 5)의 예는 7개의 unit으로 구성된 하나의 파일이 3개의 저장소에 'C→A→B→A'순으로 배치된 것을 나타낸다. 이러한 정보를 참조함으로써 하나의 파일을 구성하는 데이터들이 여러 스토리지에 어떤 형태로 분산 저장되어 있는지를 알 수 있게 된다.

RFC5661은 파일 관련 레이아웃의 세부 구조를 설명하고 있다. 한 파일에 관련된 레이아웃은 nfsv4_file_layout4와 nfsv4_1_file_layout_ds_addr로 표현된다. (그림 6)은 파일 레이아웃의 세부적인 구조를 나타낸다[14].

nfsv4_file_layout4은 파일이 스트라이핑된 DS들



(그림 5) 파일 레이아웃 개념



(그림 6) 파일 레이아웃 세부 구조

의 정보, 파일 핸들 리스트 등의 정보를 나타내며, 클라이언트에서는 LAYOUTGET을 통해 MDS로부터 얻어온다. 이 때 DS들의 정보는 레이아웃의 크기가 너무 커지지 않도록 deviceID라는 스트링 형태의 인덱스 정보로만 표현이 된다.

구체적인 DS 정보들은 nfsv4_1_file_ layout_ds_addr 레이아웃에 의해 표현이 되는데, IP 주소와 포트 번호로 구성되는 DS들의 위치 정보 및 스트라이핑 패턴 정보를 포함하며 GETDEVICEINFO를 통해 클라이언트가 MDS로부터 얻어 온다.

위의 레이아웃을 (그림 5)의 예에 대입해 보면, 저장 패턴을 표현하는 nflda_stripe_indices에는 (2,0,1,0)이라는 패턴 값이 대입되고, nflda_stripe_indices의 첫 번째 요소를 나타내는 nfl_first_stripe_index에는 0이 기록된다. 디바이스를 구성하는 DS들의 목록을 나타내는 multipath_ds_list에는 {A의 주소, B의 주소, C의 주소}가 저장된다.

DS 내에 저장되는 파일을 유일하게 식별할 수 있는 파일 핸들의 리스트를 저장하는 nfl_fh_list에는 이 레이아웃에서 설정하는 데이터 패킹 방식(sparse 모드 또는 dense 모드)에 따라 저장할 수 있는 파일 핸들의 개수 및 값이 달라진다.

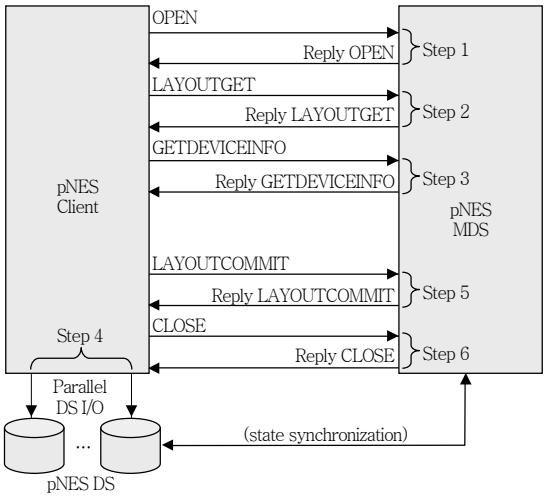
이 외에도 nfsv4_1_file_layouthint4라는 구조체를 보조적으로 사용할 수 있는데, 클라이언트가 레이아웃을 직접 설정하고 싶을 경우에 OPEN이나 SETATTR을 통해 사용된다.

3. pNFS의 동작 구조

RFC5661 표준에 의하면, pNFS와 관련한 오퍼레이션은 총 9개로 정의하고 있다. 그 중, 클라이언트가 레이아웃을 가져오고 레이아웃을 업데이트하는 오퍼레이션들은 <표 4>와 같다[14]. 나머지 4개는 back-

〈표 4〉 pNFS 오퍼레이션들

Operation	Role
GETDEVICEINFO	클라이언트가 파일이 저장된 실제 데이터 서버들에 대한 정보를 얻어오는 오퍼레이션. 클라이언트는 이 정보를 이용하여 해당 DS들에 직접 접속하여 데이터 IO 작업을 수행함.
GETDEVICELIST	클라이언트가 특정 파일 시스템에 있는 모든 device를 얻기 위해 MDS에 요청하는 오퍼레이션
LAYOUTGET	클라이언트가 파일과 관련한 레이아웃을 MDS로부터 얻어오기 위한 오퍼레이션. 파일이 저장된 실제 DS들에 대한 정보는 레이아웃에 있는 device ID를 이용해서 MDS에서 한 번 더 요청해서 얻을 수 있음.
LAYOUTCOMMIT	WRITE 수행 시 변경된 파일의 메타데이터를 클라이언트가 MDS에 알려주는 오퍼레이션임.
LAYOUTRETURN	클라이언트가 파일의 레이아웃을 반환함.



(그림 7) pNFS의 동작 방법

channel과 관련된 것으로, MDS가 클라이언트에게 위임한 레이아웃이나 lock 권한 등을 회수하기 위해 사용된다.

pNFS 오퍼레이션들을 사용한 pNFS 동작 구조를 이해하기 위해, 파일을 생성하여 WRITE하는 경우를 이용하여 설명하면 (그림 7)과 같다[1],[14]. (그림 7)에서는 세션 설정 등의 내용은 생략하였다.

- Step 1: OPEN & Reply

원하는 디렉토리에 원하는 파일을 생성하고, 클라이언트가 MDS와 파일과 관련된 메타데이터 설정을 할 수 있다. byte-range lock을 제외하는 나머지 lock의 사용 여부도 이 단계에서 설정할 수 있다.

- Step 2: LAYOUTGET & Reply
  클라이언트가 MDS에게 원하는 파일과 관련한 레이아웃을 얻기 위한 오퍼레이션을 수행하는 단계이다. 클라이언트가 레이아웃 유형, I/O 모드 등을 설정하여 MDS로 보내면, MDS에서는 device ID와 파일 핸들 리스트 등 DS 위치 정보를 클라이언트에게 알려준다.
- Step 3: GETDEVICEINFO & Reply
  Step 2에서 DS의 정보는 ID 밖에 없으므로, 클라이언트는 이 단계를 통해서 DS에 대한 IP 주소, 포트 번호 등 자세한 정보를 얻어올 수 있다. 레이아웃에 DS의 모든 정보를 넣게 되면 데이터가 너무 크므로, 이와 같이 분리된 오퍼레이션으로 정보를 제공한다.
- Step 4: Parallel DS I/O
  클라이언트가 DS에 접속하여 READ/WRITE 등 병렬 데이터 I/O를 직접 수행한다.
- Step 5: LAYOUTCOMMIT & Reply
  만약 step 4에서 WRITE 오퍼레이션을 수행하였다면, MDS는 그 파일과 관련된 정확한 메타데이터 정보를 가지고 있지 않다. 이럴 경우에는 클라이언트가 DS와의 I/O가 완료된 후 LAYOUTCOMMIT 오퍼레이션을 이용해서 업데이트 요청을 한다.
- Step 6: CLOSE & Reply
  파일에 대한 I/O가 끝났으므로, stateid 업데이트, locking 해제 등 해당 파일과 관련한 작업을 수행한다. 이 후에 LAYOUTRETURN을 사용하여 레이아웃을 MDS에 반납할 수 있다.

Ⅳ. 연구개발 현황

NFSv4.1가 RFC로 공표됨에 따라 업체들의 상품 개발뿐만 아니라 OS에서도 NFSv4.1 프로토콜을 구현하려는 노력이 활발히 이루어지고 있다. 아직 성숙 단계는 아니지만, NFS4v4.1의 기본 기능 및 pNFS의 기본 동작은 시험할 수 있도록 소스 코드가 공개되어 있다. 본 장에서는 pNFS와 관련한 연구개발 현황에 대해 OS 지원 현황과 업체별 개발 현황으로 나누어 살펴본다.

1. OS 지원 현황

pNFS가 제대로 개발되고 서비스되기 위해서 OS에서의 지원이 매우 중요하다. OS에 pNFS를 구현하는 작업은 현재 리눅스 OS와 솔라리스에서[20] 동시에 진행 중이다.

표준, 소스 코드를 포함하여 pNFS와 관련한 모든 정보는 pnfs.com 사이트에서 확인할 수 있다[21]. 또한 개발 이슈와 관련해서는 linux-nfs의 wiki 사이트에서 확인할 수 있다[22].

◎ Linux OS

리눅스 OS에서는 Panasas, NetApp, CITI 등이 주축이 되어 작업을 진행해 왔다. 크게 4단계로 나누어서 개발해 나가고 있는데, 현재는 첫 단계인 레이아웃과 관련한 동작들을 제공하는 코드를 OS에 포함시키고 있다[22]. <표 5>는 pNFS 커널 코드가 버전별로 어떤 특성을 담고 있는지 기술하고 있다[13].

현재까지 개발된 최신 버전은 리눅스 2.6.38.1 커널에 포함된 것이다[22]. 최근 pNFS 기능을 포함한

〈표 5〉 pNFS 커널 코드 구현 경과

Date	Kernel	Changes
2009. 7. 26	2.6.30	RPC sessions, NFSv4.1 server, OSDv2 rev5, EXOFS
2009. 10. 14	2.6.31	NFSv4.1 client without pNFS
2010. 2. 1	2.6.32	130 server-side patches add back-channel
2010. 6. 1	2.6.33	43 pNFS patches
2010. 7. 1	2.6.34	21 NFSv4.1 patches
2010. 9. 14	2.6.35	1 client and 1 server patch
2010. 11. 13	2.6.36	16 patches accepted into the merge
2011. 1. 6	2.6.37	Includes first chunk of pNFS beyond generic 4.1 infrastructure
–	2.6.×× or 3.×	- 250 patches of remaining pNFS divided into 4 waves - Remaining file-based pNFS patches - object-based pNFS - Block-based pNFS

〈표 6〉 pNFS 소스코드 형상

Branch	Description
Nfs41	Session and other ono-pnfs NFSv4.1 code
pnfsd	nfs41, pnfsd, pnfsd-exofs, pnfsd-lexp, pnfs, pnfs-obj, pnfs-block, spnfs 모두 포함.
pnfsd-exofs	EXOFS를 위한 Objec-based pNFS server extension
pnfsd-lexp	pNFS Local Export of any local file system
pnfs-obj	Object-based layout driver
pnfs-block	Block-based layout driver
spnfs	Simple pNFS server. NetApp에서 개발해 왔는데, 현재는 지원 안됨. pnfsd에 포함.

Fedora RPM 버전을 제공하는 사이트도 있다[23]. 소스 코드 다운로드와 설치 방법에 관해서는 linux-nfs 사이트를 참고할 수 있다[22]. 또한 pNFS 소스들은 linux-nfs git 리파지토리 트리에 저장되어 있는데, 그 형상은 <표 6>과 같다[13],[22].

pNFS를 포함하여 NFSv4.1 코드 구현과 관련한 이슈들은 wiki 사이트에서 확인할 수 있는데, cluster coherent NFS 이슈 및 그와 관련한 locking 이슈, NFS 서버 및 클라이언트 복구, migration, delegation, SPKM3 등 보안 관련 사항들이 주요 이슈들로 논의되고 있다[23].

2. 업체별 개발 현황

가. Panasas

Panasas는 지금까지 pNFS 표준화 및 구현에 가장 큰 기여를 해 오고 있다. 또한 자사 제품인 PanFS는 객체 기반으로 구현된 병렬 파일 시스템으로서, 현재는 pNFS와 호환 가능하다[5].

나. NetApp

NetApp의 pNFS 구현은 Data ONTAP GX/Striped WAFL 대상으로 이루어졌다. 리눅스 OS에 pNFS 기능을 구현하는데도 적극적이었으며, spNFS (simple pNFS)라는 파일 기반의 pNFS 기능을 구현하여 공개하기도 했다[24]. 그러나 2010년에 linux-nfs 작업에서 탈퇴하였고, spNFS는 리눅스 pNFS 커널 개발자 코드에 포함되었다. 현재 NetApp은 IETF의 nfsv4 WG에서 FedFS(Federated File System) 표준화 부분에 기여하고 있다[25].

다. Lustre

대용량 병렬 파일 시스템인 Lustre를 만든 Cluster File Systems사가 Sun에 인수되었고, Sun은 다시 오라클에 인수되었다. 2010년에 오라클이 Lustre 개발 중단을 공표하는 바람에 Lustre는 개발 일정이 불투명해졌다[7]. 현재 Lustre는 원래의 Lustre 개발자들이 설립한 Whamcloud사에 의해 오픈 소스 형태로 개발이 계속 진행 중이다. Whamcloud사가 2011년 6월에 발표한 Lustre 개발 로드맵에 의하면 2013년 정도에 pNFS 기능이 추가될 예정이다[26].

라. IBM

IBM은 클러스터 파일 시스템인 GPFS를 연계하여 pNFS를 구현하는 프로젝트를 진행하고 있다[27].

IBM은 고성능/고확장 NAS(Network Attached Storage) 시스템을 구현하는 목표를 달성하기 위해 pNFS와 GPFS를 이용하고 있다.

마. EMC

EMC는 pNFS가 블록 기반의 파일 시스템을 지원할 수 있도록 레이아웃 표준화 및 드라이브 구현 작업을 진행하였고, 현재도 pNFS 와 관련한 활동에 활발히 참여하고 있다[6].

바. BlueArc

BlueArc는 최근 2년 동안 SC conference에서 pNFS 구현 결과를 데모하며 활발한 활동을 하고 있다. BlueArc에서 개발한 스토리지는 객체 기반의 다중 프로토콜을 지원하는 구조를 가진다[7].

V. 결론

지금까지 NFSv4.1의 표준화 및 업체 개발 현황에 대해 살펴보았다. 현재까지는 구현 중이어서 완성되어 사용되고 있는 제품은 거의 없다. 그러나 Panasas가 그들의 제품인 PanFS에 pNFS 기능을 제공하기로 방침을 정한 것과 같이 관련 업체들은 자체 제품에 pNFS 기능 구현을 염두에 두고 있으므로, 머지 않은 미래에 기존의 NFS의 기능을 완전히 대체할 것이라 기대된다. 또한 오픈 소스인 Lustre에서도 로드맵에 pNFS 구현을 포함시키고 있는 등 pNFS의 활성화는 시간 문제인 것으로 기대된다.

기타, pNFS 표준화와 관련하여 추후 논의될 수 있을 것으로 예상되는 이슈 중에는, metadata clustering 이슈가 있다[28]. 현재, NFSv4.1에서는 한 개의 MDS를 가정하고 있고, 클러스터링은 자체적으로 해결하는 것으로 되어 있다. 그러나 성능 이슈로 인해 추후 메타데이터 처리가 병목점이 될 가능성이 높으므로, 조만간 표준화 이슈가 될 것으로 판단된다.

● 용 어 해 설 ●

NFSv4.1: 2010년에 IETF에서 제정한 NFS의 새로운 표준. 기존 NFS의 제한점들을 해결하기 위해 시큐리티, 세션, 데이터 병렬 액세스, 클라이언트로의 권한 위임 등의 특성들을 추가함.

pNFS: NFSv4.1에서 성능과 확장성 지원을 위해 추가한 데이터 병렬 액세스 특성

약어 정리

DS	Data Server
FedFS	Federated File System
GFS	Global File System
GPFS	General Parallel File System
GSS	Generic Security Service
HPC	High Performance Computing
IB	InfiniBand
IETF	Internet Engineering Task Force
MDS	MetaData Server
NAS	Network Attached Storage
NFS	Network File System
OSD	Object Storage Device
pNFS	parallel NFS
RDMA	Remote Data Memory Access
SPKM	Simple Public-key GSS-API Mechanism
spNFS	simple pNFS
XDR	External Data Representation

참 고 문 헌

[1] Thijs Stuurman, “pNFS: Parallel Network File System, Universiteit Van Amsterdam,” June 2008.

[2] Dean Hildebrand and Peter Honeyman, “Direct-pNFS: Scalable, Transparent, and Versatile Access to Parallel File Systems,” *HPDC*, 2007.

[3] Dean Hildebrand, Peter Honeyman, and Andy Adamson, “pNFS and Linux: Working towards

a Heterogeneous Future," CITI.
[4] Robin Harris, "Parallel NFS: Finally, NFS Optimized for Clusters," Data Mobility Group, Mar. 2007.
[5] Garth Gibson, "Parallel NFS(pNFS)," Panasas, 2007.
[6] Spencer Shelper, "pNFS(Parallel NFS) BOF," *SC*, Nov. 2008.
[7] Addison Snell, "The State of pNFS: The Parallel File System Market in 2011," Intersect360 Research, Mar. 2011.
[8] IETF NFSv4 WG. http://www.ietf.org.
[9] J. Haynes and D. Noveck, "Network File System (NFS) Version 4 Protocol," RFC3530, IETF, 2003.
[10] Alex McDonald, "Migration from NFSv3 to NFSv4," SNIA, Apr. 2011.
[11] Vincent Roqueta, NFSv3 to NFSv4 Migration Guide. http://nfsv4.bullopensource.org/doc/NFS 3_NFS4_migration.pdf
[12] CITI(Center of Information Technology Integration). http://www.citi.umich.edu
[13] Brent Welch, "pNFS Update: A Standard for Parallel Ffile Systems," HPC Advisory Council, Mar. 2011.
[14] D. Black, S. Fridella, and J. Glasgow, "Network File System (NFS) Version 4 Minor Version 1 Protocol," RFC5661, IETF, 2010.
[15] Lars Eggert, "Network File System (NFS) Version 4 Minor Version 1 External Data Representation (XDR) Description," RFC5662, IETF, 2010.
[16] D. Black, S. Fridella, and J. Glasgow, "Parallel NFS (pNFS) Block/Volume Layout," RFC5663, IETF, 2010.
[17] B. Halevy, B. Welch, and J. Zelenka, "Object-based Parallel NFS (pNFS) Operations," RFC 5664, IETF, 2010.
[18] Referral, Migration and Replication for NFSv4 User Guide, GNU/Linux NFSv4 Project. http://nfsv4.bullopesource.org/doc/migration-and-replication-0.2.pdf
[19] Garth Goodson, Sai Susarla, and Rahul Iyer, Standardizing Storage Clusters: Will pNFS become the new standard for parallel data access?, 2007. http://potal.acm.org/ft_gateway.cfm?id=1317402&type=pdf
[20] Open Solaris website. http://www.opensolaris.org
[21] pNFS website. http://www.pnfs.com
[22] Linux-nfs development site. http://www.linux-nfs.org
[23] Linux-nfs wiki page. http://wiki.linux-nfs.org/wiki/index.php/Main_Page
[24] Dan Muntz, Mike Sager, and Ricardo Labiaga, spNFS: A Simple pNFS Server, 2008. http://www.con-nectathon.org/talks08/dmuntz-spnfs-ctho n08.pdf
[25] J. Lentini, "Requirements for Federated File Systems," RFC5716, IETF, 2010.
[26] Eric Barton, "Lustre pNFS Development Roadmap," HPC Advisory Council, 2011.
[27] IBM Almaden, pNFS Project Description. http://www.almaden.ibm.com/storagesystems/projects/pnfs
[28] Architecture – Clustered Metadata. http://wiki.lustre.org