# NFS 4.1 (pNFS)

Falko Benthin, f.benthin@jpberlin.de, s0522617

1. Juni 2011

## Inhaltsverzeichnis



# 1 Einführung

Mit der zunehmenden Digitalisierung aller Lebensbereiche hat sich das Datenaufkommen in den letzten Jahren stark erhöht. Es fallen nicht nur mehr Daten an, sondern sie müssen auch verwaltet und so abgespeichert werden, dass von verschiedenen Stellen darauf zugegriffen werden kann. Hier helfen verschiedene Netzwerkspeicherlösungen, von denen sich vor allem das Network File System (NFS) großer Beliebtheit erfreut. Es wurde in den 80er Jahren des vergangenen Jahrhunderts entwickelt und gilt als ältestes Protokoll für Networked Attached Storage (NAS). Es unterstützt eine große Bandbreite von Netzwerkhardware (Ethernet, Fibre Channel, ...) und wurde für verschiedene Plattformen implementiert, so dass es preiswerte NAS-Lösungen ermöglicht. Mittlerweile wird NFS von der Internet Engineering Task Force (IETF) entwickelt und ist bei der Version 4.1 angelangt. Die Version 4.2 befindet sich in Entwicklung.

Bis zur Version 4 ließ sich NFS nur als Single-Server-Lösung einsetzen. Solange das Datenaufkommen nicht hoch ist, mögen die Datendurchsatzraten akzeptabel sein, aber im High-Performance-Computing (HPC) erwies sich NFS als Flaschenhals, der das gesamte System ausbremste oder andere, deutlich kostspieligere Lösungen erforderlich machte. In der Spezifikation von NFS 4.1 wurde das pNFS-Protokoll eingeführt, dass die bekannten Performanceprobleme beseitigt und parallele Datenübertragungen erlaubt.

# 2 Netzwerkspeicherlösungen

Der steigende Bedarf nach zentralen Speicherlösungen und deren Verwaltung hat dazu geführt, dass neben NFS weitere Netzwerkdateiprotokolle entwickelt wurden. Es konnten sich jedoch nur wenige durchsetzen. Ein Netzwerkprotokoll erleichtert Anwendern und Administratoren die zentrale Datenhaltung:

- Daten lassen sich leichter sichern. Zentral gehalten muss nur noch ein Backup angelegt werden, anstelle mehrerer lokaler Sicherungen.
- Die Daten sind an vielen Stellen verfügbar, so dass Nutzer verschiedener Standorte auf sie zugreifen können.
- Die Datenspeicherung erfolgt bei vielen Lösungen transparent. Dank des Netzwerkdateiprotokolls ist es egal, welche Dateisystem letztendlich für die Datenspeicherung genutzt wird.
- Zugriffsrechte können zentral verwaltet werden.

Es ist für kostenbewusste Anwender von Vorteil, wenn ihre Netzwerkspeicherlösung preiswerte Standardhardware unterstützt, so dass sie bei Bedarf ohne hohen finanziellen Aufwand und herstellerunabhängig erweiterbar ist.

## 2.1 Überblick Netzerkdateisysteme

Zu den aktuell weit verbreiteten Netzwerkdateisystemen zählen die folgenden:

- NFS: ist das älteste Protokoll für Netzwerkdateisysteme. Es wurde bereits 1985 von Sun entwickelt und basiert auf IP, TCP (bis Version 3 auch UDP) und RCP. NFS ab Version 4 ist von der IETF standardisiert, viele Hersteller beteiligen sich an der Entwicklung und es ist weit verbreitet. Aktuelle NFS-Versionen unterstützen Access Control Lists (ACLs) und Kerberos-Authentifizierung. NFS kann jedes Dateisystem exportieren, dass vom dateiserver unterstützt wird.
- Common Internet File System (CIFS): ist ein 1996 von Microsoft entwickeltes Protokoll. CIFS ist für viele Betriebssysteme verfügbar und zu großen Teilen dokumentiert. Es bietet einen breiteren Funktionsumfang als ähnliche Protokolle (darunter Druckdienste und ACLs) sowei eine gute Performance. Es wurde nie von der IETF standardisiert.
- WebDAV: ist ein offizieller Standard, um über das Internet auf Dateien zuzugreifen und weit verbreitet. Das WebDAV-Protokoll hat einen hohen Overhead und eine langsame Datenübertragung.

- Andrew Filesystem (AFS): ist weit verbreitet und Clients sind für alle gängigen Betriebssysteme verfügbar. Es bietet eine gute Kerberos-Integration, lässt sich gut skalieren und erfordert einen Datenbankserver.

- Lustre: ist ein freies Cluster-Filesystem für HPC, das parallele Datenzugriff über das Netzwerk erlaubt. Lustre wird von Oracle (ehemals Sun) als Open-Source-Projekt entwickelt. In der Praxis ist es mangels Unterstützung für verschiedene Betriebssysteme nicht sehr verbreitet.

- Apple Filing Protocol (AFP): ist ein von Apple entwickeltes Netzwerkdateisystem, dass vorrangig mit Mac OS X eingesetzt wird.

- General Parallel File System (GPFS): stabiles, von IBM entwickeltes Cluster-File-System für Systeme mit Linux und AIX.

## 2.2 Probleme und Beschränkungen bei Single-Server-Lsg.

Viele der oben genannten Netzwerkdateisysteme haben gemeinsam, dass es sich um Single-Server-Lösungen handelt. Ein einzelner Server ist sowohl dafür zuständig, Anfragen nach Dateien entgegen zu nehmen als auch dafür, sie zu speichern und auszuliefern.

Vor allem im Bereich HPC kommt es so zu Performanceproblemen, da die Daten nicht schnell genug ausgeliefert werden. Werden viele kleine Dateien angefragt, wird übermäßig viel Zeit verbraucht, um die Daten zusammenzustellen und zu versenden. Bei großen Dateien werden vor allem Begrenzungen bzgl. der Bandbreite gesetzt. Auch die begrenzten Speicherkapazitäten einzelner Server haben sich in der Vergangenheit als Problem erwiesen.

## 2.3 Abhilfen inkl. Vor- und Nachteile

Um den oben genannten Problemen abzuhelfen, wurden mehrere Lösungen entwickelt [GSI08]:

- Partitionierung der Daten: Daten werden über mehrere Fileserver verteilt. Da die Partitionierung manuell erfolgt, ist die Wartung kompliziert, da auf jedem Clients mehrere Mount-Points angelegt werden müssen. Sobald ein Server ausfällt, kann es passieren, dass die Anwendungen unbenutzbar werden, die auf dessen Daten angewiesen sind. Das Problem des begrenzten Speicherplatzes auf einem einzelnen Server besteht immer noch, zudem werden Administratoren und Anwender mit Herausforderungen konfrontiert, wenn der Speicherplatz einzelner Partitionen ausgeschöpft ist.

- Transarentes Routing: Die Datenspeicherung wird mit Hilfe einer zusätzlichen Routing-Schicht vor den Anwendungen verborgen. Dadurch ist es möglich, neue Clients ohne große Anpassungen ins System einzubinden. Problematisch ist immer noch die Performance, da alle Daten den Rechner passieren müssen, der für das transparente Routing verantwortlich ist.

- Paralleler Speicherzugriff: Die erfolgsversprechendste Lösung hinsichtlich Performance und Speicherplatz bietet der parallele Speicherzugriff. Daten können parallel geladen und abgelegt werden, wodurch sich Flaschenhälse vermeiden lassen. Verschiedene Hersteller haben Lösungen für den parallelen Speicherzugriff entwickelt, sich jedoch nicht auf ein standardisiertes Protokoll geeinigt, so dass viele Lösungen proprietär sind und ggf. Anpassungen der Anwendungen erforderlich machen. Die Lösungen einzelner Hersteller funktionieren meist nur mit deren Clients zufriedenstellend, die aber oft nicht für alle Betriebssysteme eines heterogenen Netzwerks oder mehrere Speichersysteme vorliegen.

Das in NFS 4.1 enthaltene Protokoll pNFS unterstützt den letzten Lösungsansatz. Da es sich um einen offenen Standard handelt, der sich zudem noch leicht in Betrieb lässt, steht einer weiten Verbreitung nichts entgegen.

# 3 Überblick pNFS

## 3.1 Motivation

pNFS soll die Lücke zwischen den bisherigen NFS-Protokollen der Versionen 2, 3 und 4 sowie parallelen Cluster-Dateisystemen schließen. Die bisherigen NFS-Versionen ließen sich nur im Single-Node-Modus betreiben, so dass es hier im HPC zu den schon erwähnten Engpässen kam. Dank pNFS können Clients direkt auf Daten zugreifen, die auf mehreren Fileservern verteilt liegen und dabei im günstigsten Fall die volle Bandbreite ausnutzen, die das Netzwerk bereitstellt (siehe Abbildung 1). Die Arbeit an pNFS begann bereits 2006, die Spezifikationen wurden 2010 als RFC 5661 von der IETF verabschiedet [Int10a].

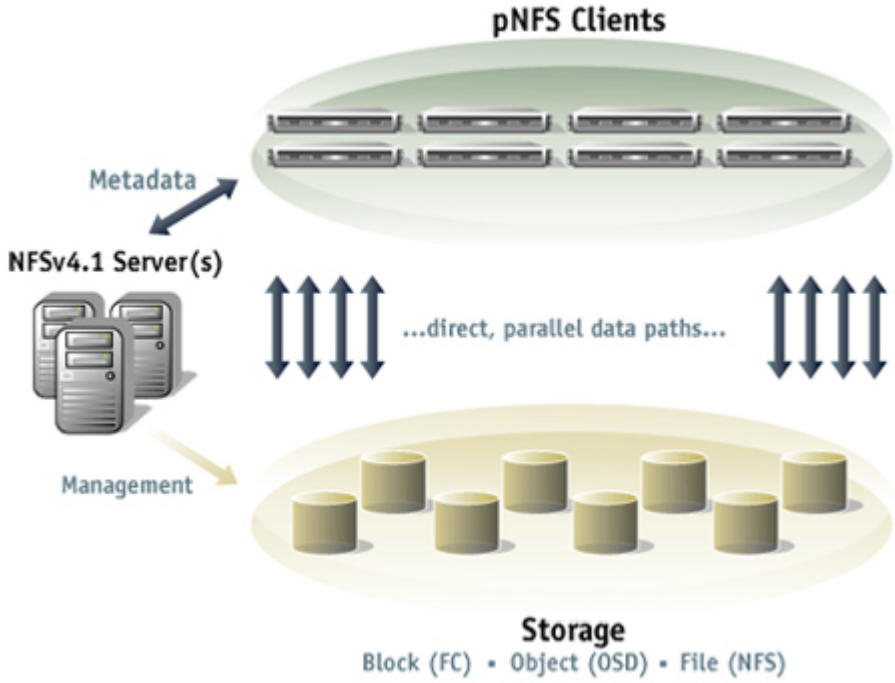


**Abbildung 1:** pNFS erlaubt direkten Zugriff auf mehrere Fileserver ©Panasas

## 3.2 Vorteile

Da es sich bei pNFS um eine standardisierte Erweiterung für NFSv4 handelt, wird die Administration der Clients erleichtert. Aufgrund der Standardisierung ist NFS für viele Betriebssysteme verfügbar, so dass es auch für heterogene Netzwerke geeignet ist. pNFS ist abwärtskompatibel. Dank der daraus resultierenden Interoperabilität zwischen alten und neuen Clients müssen nicht alle Clients eines Netzwerks umgerüstet werden, wenn ein pNFS-File-Server-Cluster in Betrieb genommen wird. Da pNFS-Cluster wie lokale Laufwerke gemountet (eingehangen) werden, können viele Anwendungen ohne weitere Anpassungen auf die Daten des Netzwerkspeichers zugreifen.

Das pNFS-Protokoll erlaubt eine skalierbare Anpassung der Fileserver. Sobald der Datendurchsatz einen kritischen Wert unterschreitet, können Administratoren dem Cluster ohne großen Aufwand weitere Nodes hinzufügen um die Performance zu steigern. Da Daten über mehrere Server verteilt werden dürfen, gehören auch die Speicherplatzbeschränkungen der alten Versionen der Vergangenheit an.

Da pNFS von vielen großen Herstellern (z.B. IBM, Panasas, Microsoft, Network Appliance, EMC, und Oracle (ehemals Sun)) unterstützt und weiter entwickelt wird, dürfte es schnell auf breite Akzeptanz stoßen. Zudem lässt sich pNFS mit verschiedenen Netzwerkspeichertechnologien (Fileserver, SAN, ...) einsetzen. Ein weiterer Vorteil ist, dass nur noch ein Port in Firewalls geöffnet werden muss, um auf NFS-Server zuzugreifen.

### 3.3 Voraussetzungen

Um pNFS zu nutzen, muss es vom Kernel des Betriebssystems unterstützt werden. Bei Linux hängt es vom Distributor ab, ob pNFS aktiviert ist. In der Regel sollte es bei Distributionen mit einem Kernel ab 2.6.36 der Fall sein. Wer den Kernel selbst kompiliert, muss darauf achten, die entsprechenden Parameter zu setzen (CONFIG_NETWORK_FILESYSTEMS, CONFIG_NFS_FS, CONFIG_NFS_V4, CONFIG_NFS_V4_1, CONFIG_PNFS, CONFIG_NFSD, CONFIG_PNFSD, CONFIG_PNFSD_LOCAL_EX CONFIG_PNFS_FILE_LAYOUT). Clients sollten für alle gängigen Betriebssysteme verfügbar sein.

## 4 Funktionsweise

pNFS basiert auf einer Client-Server-Architektur. Der Server besteht aus zwei Komponenten, dem Metadaten-Server und den Daten-Servern. Der Metadatenserver enthält in den sogenannten Layouts alle Informationen, wo einzelne Dateien auf den Datenservern zu finden sind, wie groß sie sind und welche Dateiattribute gesetzt wurden. Ein Client stellt eine Anfrage an den Metadatenserver. Dieser prüft, ob der Client Zugriffsrechte hat und liefert ihm ggf. das entsprechende Layout, so dass der Client anschließend in der Lage ist, sich direkt mit (mehreren) Datenservern zu verbinden und die benötigten Daten zu laden oder Daten zu schreiben. Anschließend ändert er Metadaten auf dem Metadaten-Server, z.B. ctime, atime und mtime, Größe der geschriebenen Blöcke etc., wartet auf die Bestätigung vom Metadatenserver und schließt die geöffnete Layoutdatei (siehe Abbildung 2).

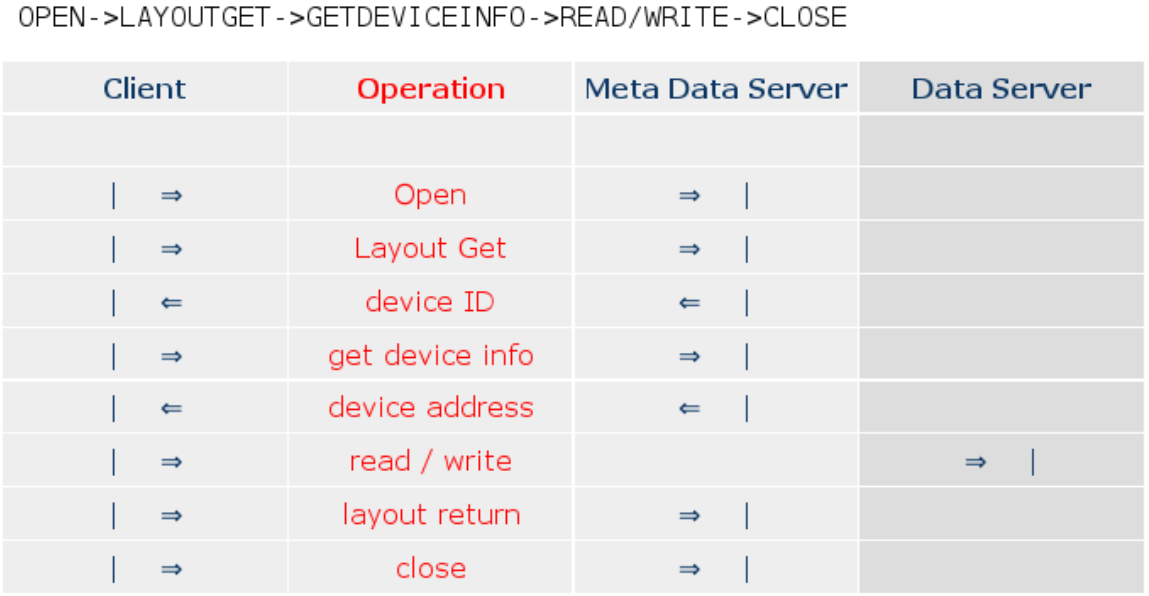


**Abbildung 2:** I/O-Operationen bei Dateizugriff (File-Layout) ©dCache.org

Die pNFS-Architektur basiert auf drei Prinzipien:

- Vertraute Semantik: die Semantik des pNFS-Protokolls ähnelt der des NFSv4-Protokolls, wodurch es leichter ist, pNFS bestehenden NFS-Anwendungen zugänglich zu machen.

- Einfachheit: Die pNFS-Erweiterung muss einfach gehalten werden. Ihre Schnittstelle muss die grundlegenden Funktionalitäten bereitstellen, die für Parallelität erforderlich sind und darüber hinaus zusätzliche Ebenen für zusätzliche, komplexere Semantik erlauben.

- Flexibilität: pNFS muss flexibel genug sein, um mit verschiedenen Speicher-Architekturen (z.B. Stand-Alone-File-Server, NAS und RAID-Geräte) zu funktionieren und diese zu einem Cluster-File-System zusammenfassen können.

## 4.1 pNFS-Protokoll

Das pNFS-Protokoll ist in RFC 5661 [Int10a] beschrieben. Es arbeitet mit mehreren anderen Protokollen zusammen, dem Kontroll-Protokoll (Control Protocol) und dem Speicher-Zugriffs-Protokoll (Storage Access Protocol) (siehe Abbildung 3). Mittels pNFS-Protokoll greift ein Client auf den Metadatenserver zu. Das Speicher-Zugriffs-Protokoll dient der Kommunikation zwischen Client und Daten-Servern und mit dem Kontroll-Protokoll werden die Daten zwischen Metadaten- und Datenservern abgeglichen.

In pNFS wird jede Datei mittels einer zwölfstelligen pnfsID referenziert, die mit den iNodes anderer Dateisysteme vergleichbar ist. Eine pnfsID, die einmal benutzt wurde, wird niemals erneut vergeben, selbst dann nicht, wenn die zugehörige Datei bereits gelöscht wurde.

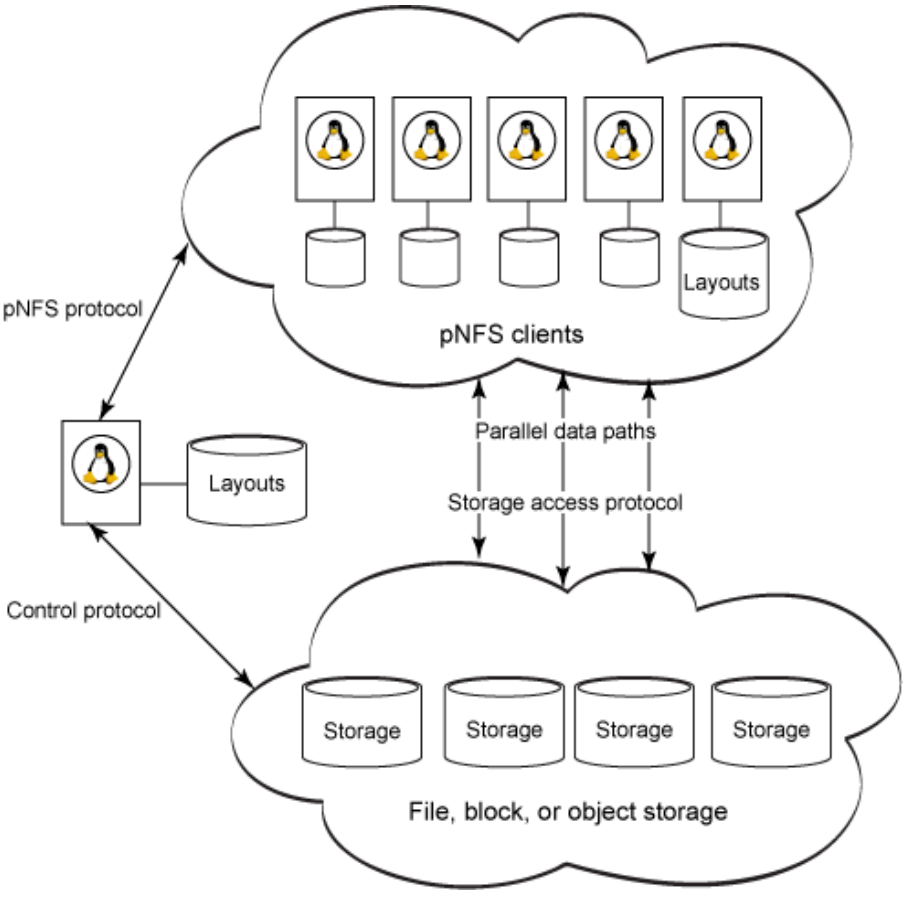


**Abbildung 3:** Protokolle in einem pNFS-Cluster ©IBM

Das pNFS-Protokoll erweitert NFSv4 um sechs Operationen. Vier davon sind für Layouts zuständig (siehe unten) und zwei betreffen die Namen für Datenserver. Da die Namen für Datenserver sehr komplex werden können und mitunter mehrere Kilobytes lang werden, verfügt das pNFS-Protokoll über Mapping-Funktionalitäten. Komplexe Namen werden durch kurze Data-Server-IDs ersetzt, die in den Layouts referenziert werden.

Dank der Trennung von Metadaten und Daten werden nicht nur parallele Zugriffe auf Datenserver erlaubt. Durch die Speicher-Zugriffs-Protokoll ist es auch möglich, eine große Zahl verschiedener Datenspeichergeräte, auch nicht-NFS-Datenserver, anzusprechen. In einem Cluster können sich neben mehreren Datenserver auch mehrere Metadatenserver befinden, die die selben Daten exportieren. Anwender können ein beliebiges Protokoll verwenden, um die Daten der Metadatenserver untereinander zu synchronisieren und konsistent zu halten.

## 4.2 Layout-Typen

Ein pNFS-Layout enthält alle Informationen, die ein Client benötigt um Dateien vom Datenserver abzurufen. Dazu gehört, wo eine Datei liegt, wie sie direkt abgerufen werden kann und welches Storage-Access-Protokoll (NFS, iSCSI, ...) zu verwenden ist. Alle Informationen werden in einer Gnu DBM-Datenbank gespeichert.

Ein Datenserver kann eine Datei komplett oder teilweise bereitstellen. Aus Performancegründen ist es möglich, dass Clients Layouts zwischenspeichern. Daher ist es nötig, dass der Metadatenserver ein Layout widerruft, wenn eine Datei geändert oder verschoben wird. Um die Konsistenz der Datenbestände zu gewährleisten, muss der Widerruf erfolgen bevor ein Layout geändert wird. Sobald ein Layout widerrufen wurde, können Clients nicht mehr direkt auf die Datenserver zugreifen, sondern müssen ein neues Layout anfordern. Die oben erwähnten Protokolloperationen für Layouts dienen dem Empfangen (get), Zurückgeben (return), Widerrufen (recall) und Bestätigen von Änderungen (commit changes) von Metadaten für einzelne Dateien.

Aktuell werden abhängig vom verwendeten Speicher-Zugriffs-Protokoll drei Layouttypen unterschieden:

- file: das File-Layout wird genutzt, wenn Dateien über einen oder mehrere Server verteilt sind und mittels in NFSv4 spezifizierten Lese- und Schreiboperationen angesprochen werden. Im Layout ist hinterlegt, welche Dateistücke auf welchem Server zu finden sind

- object: Objekt-Layouts ([Int10c]) ähneln File-Layouts. Sie nutzen SCSI-Objekt-Kommandos, um Daten auf Object-based Storage Devices (OSDs) zu lesen und zu schreiben. Anstelle von Dateinamen werden Objekt-IDs verwendet. Objekt-Layouts enthalten mehr Informationen als Datei-Layouts und erlauben so beispielsweise eine feinere Rechte- und Zugriffsverwaltung.

- block: ein Block-Layout ([Int10b]) exportiert in einem Storage Area Network (SAN) gehostete Logical Unit Numbers (LUNs), sprich Festplatten- oder Raid-Arrays und nutzt iSCSI- oder Fibre Channel, um auf die Speichergeräte zuzugreifen. Im Block-Layout sind die physischen Speicherblöcke für jeden Teil einer Datei hinterlegt. Wird eine Datei größer, wächst auch die Layout-Datei.

Um die Performance zu steigern, können Clients Layouts im Cache vorhalten, bis sie nicht mehr benötigt oder vom Metadatenserver widerrufen wwerden.

pNFS unterstützt eine Close-to-Open-Semantik. Wenn mehrere Clients die gleiche Datei geöffnet haben, werden Änderungen erst sichtbar, nachdem die Datei geschlossen und wieder geöffnet wurde. Um die Konsistenz zu gewährleisten, können Clients beim Metadatenserver einen NFSv4 Byte-Range-Lock beziehen, wodurch gleichzeitige Updates serialisiert werden. Es ist nicht möglich, einzelne I/O-Operationen auf mehreren Servern gleichzeitig auszuführen. Anwender, die solche ein Feature benötigen, müssen es selbst implementieren.

### 4.3 Sicherheit und Zugriffsrechte

Bezüglich Sicherheit und Zugriffsrechte greift pNFS auf die NFSv4-Operationen zurück (Open, Lock). Die Kontrollen werden, soweit möglich, bei allen I/O-Operationen durchgeführt. Diese werden jedoch nicht von allen Speicher-Zugriffs-Protokollen unterstützt. Beim Aufsetzen eines pNFS-Clusters ist darauf zu achten, dass mit den unterschiedlichen Layouts auch unterschiedliche Authentifizierungs- und Sicherheitsmodelle einhergehen.

Beispielsweise ist es so, dass derartige Prüfungen bei Block-Geräten aufgrund der eingeschränkten Schnittstellen (z.B. Fibre Channel) unmöglich vom NFS-Server durchgeführt werden können und so vom Server in den Client verlagert werden. Die Granularität der Zugriffe bei Blockgeräten beschränkt sich in der Regel auf einzelne LUNs ([GSI08] und [Int10b]).

Bei File- und Object-Layouts lassen sich dagegen Zugriffsrechte für einzelne Dateien oder Objekte vergeben. Beim Object-Layout werden die Rechte im Metadatenserver generiert und von den Datenservern geprüft ([GSI08] und [Int10c]). Die File-Layouts nutzen die bereits von NFSv4 bekannten ACLs für die Dateirechte. Sie werden ebenfalls im Metadatenserver gesetzt.

## 5 Bezug zum Grid-Computing (dCache.org

Der Einsatz von pNFS in der Forschung wird seit mehreren Jahren von mehreren Institutionen evaluiert, darunter das Deutsches Elektronen Synchrotron (DESY) [DES11] und European Organisation for Nuclear Research (CERN) [CER11]. In den beiden Forschungseinrichtungen, die sich mitunter mit Atomphysik beschäftigen, fallen mitunter mehrere Petabyte Daten an, die verarbeitet werden müssen. Beispielsweise produziert der Teilchenbeschleuniger des CERN einen Datenstream mit 300 MB/s. Sie sind darauf angewiesen, dass ihre Netzwerkspeicherlösungen die benötigten Daten schnell bereitstellen bzw. speichern können.

Hier kam es in der Vergangenheit oft zu den Problemen, dass die Speicherkapazitäten einzelnder Festplatten bzw. Server nicht ausreichen, so dass die Einrichtungen seit 1989 mit verschiedenen Lösungen experimentierten und eigene Lösungen implementierten, etwa Castor, rfio, xrootd, DPM, Storm und dCache. Seit 2006 prüfen sie, ob pNFS ihren Anforderungen genügt.

Momentan gibt es mehrere pNFS-Server: Linux, NetApp, Oracle (Sun), DPM, Storm und dCache. dCache ist eine Storage-Distribution [dCa11], die von mehreren Forschungseinrichtungen gemeinsam entwickelt wird (DESY, Fermi National Accelerator Laboratory (FNAL) [FNA11] und Nordic Data Grid Facility (NDGF) [NDG11]). Aktuell (dCache 1.9.12) unterstützt sie nur das File-Layout und kann die Dateien noch nicht über mehrere Server verteilen. dCache.org stellt für Scientific Linux 5 (SL5), ein Red Hat Enterprise Linux-Derivat, einen pnfs-fähigen Kernel bereit.

Alle Evaluierungen von pNFS am DESY durch dCache.org im letzten Jahr verliefen durchweg befriedigend und wurden den Erwartungen gerecht. Als Testumgebung fungierten ein Metadatenserver (Headnode, 4 core, 8 GB RAM, 1 GB/s Uplink), fünf NFS-Server (8 cores, 12 GB RAM, 12x2TB SATA, 80TB, 10 GB/s uplink) und 32 pNFS-fähige Clients (8 core, 16GB RAM, SL5.3, Kernel 2.6.37-pnfs) 4. Der Test lief über 10 Tage, während dieser Zeit wurden 13 TB durch Netzwerk bewegt und es kam zu keinem einzigen Absturz. Die Forscher waren lediglich an der Performance des pNFS-Protokolls interessiert, so dass alle sequentiell empfangenen Daten, einzelne Dateien mit einer Größe von jeweils 100 GB, via „cat <filename> /dev/null“ sofort wieder entsorgt wurden.

Wie die Auswertung der Tests zeigt, kommt pNFS auf ähnliche Werte wie das proprietäre Protokoll dCap (Abbildungen 5 und 6).

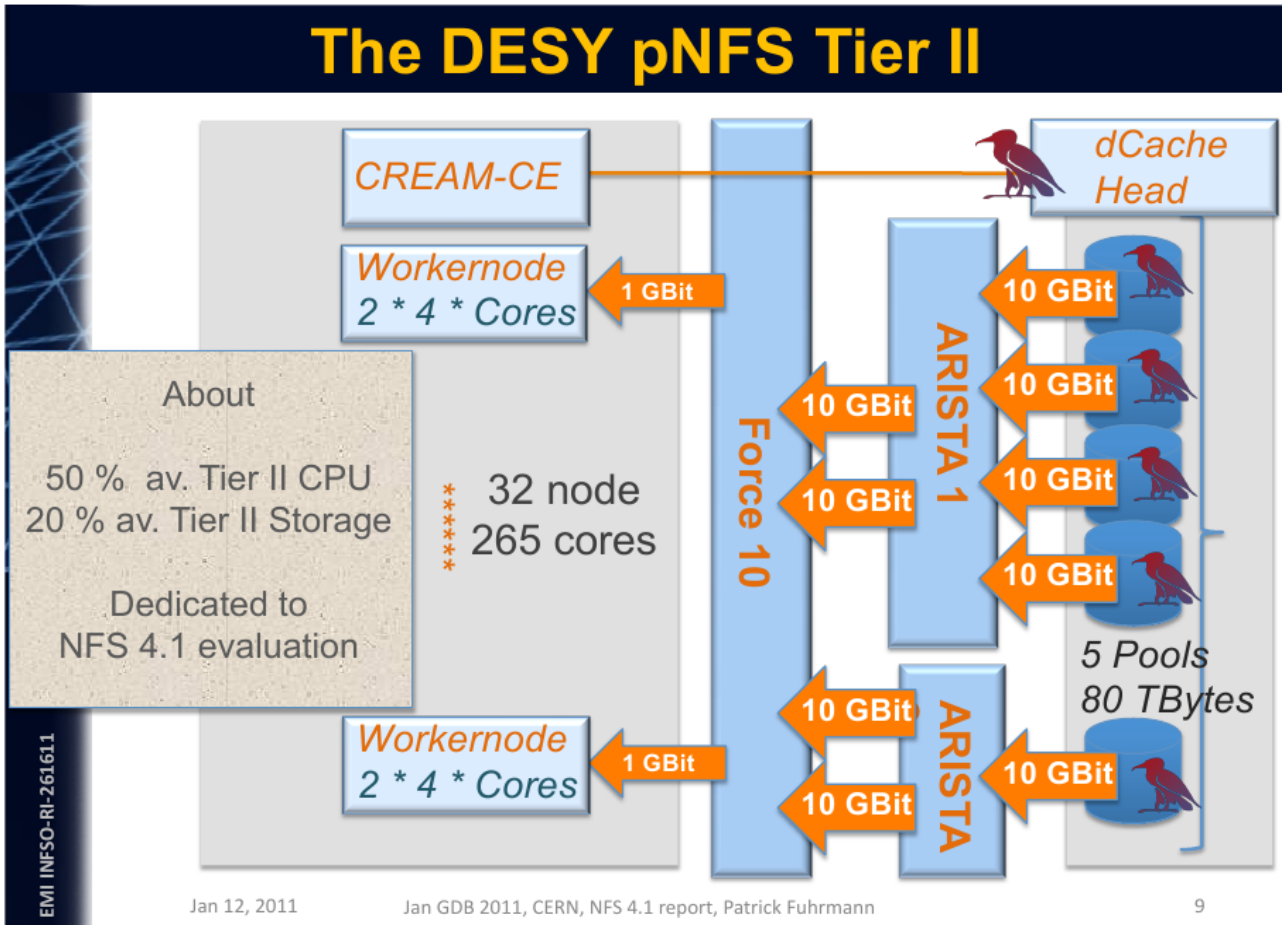


**Abbildung 4:** pNFS-Testumgebung am DESY
©CERN, NFS 4.1 report, Patrick Fuhrmann

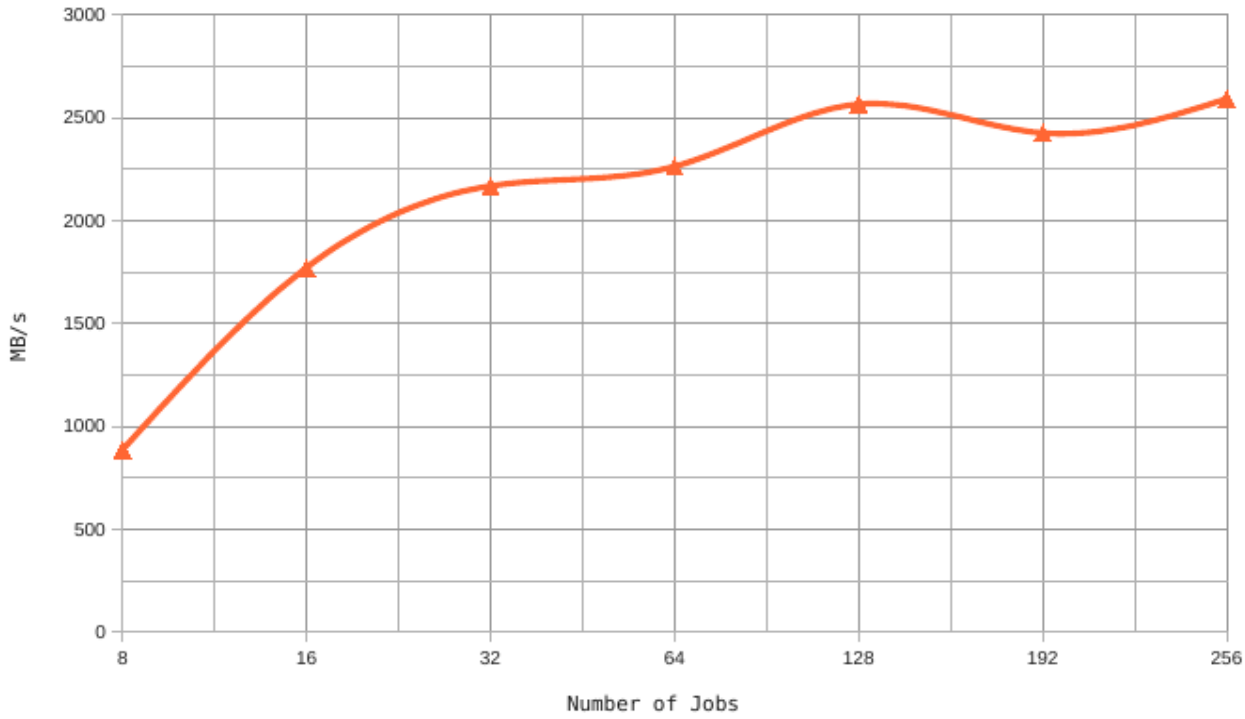


**Abbildung 5:** Network bandwith vs. number of jobs
©Tigran Mkrtchyan, pNFS under fire

dCache.org empfiehlt die Nutzung von pNFS aufgrund der Herstellerunabhängigkeit und der Standardisierung RFC 5661, weil Applikationen ohne Anpassung mit pNFS funktionieren und der pNFS-Client bereits im Betriebssystem enthalten ist.

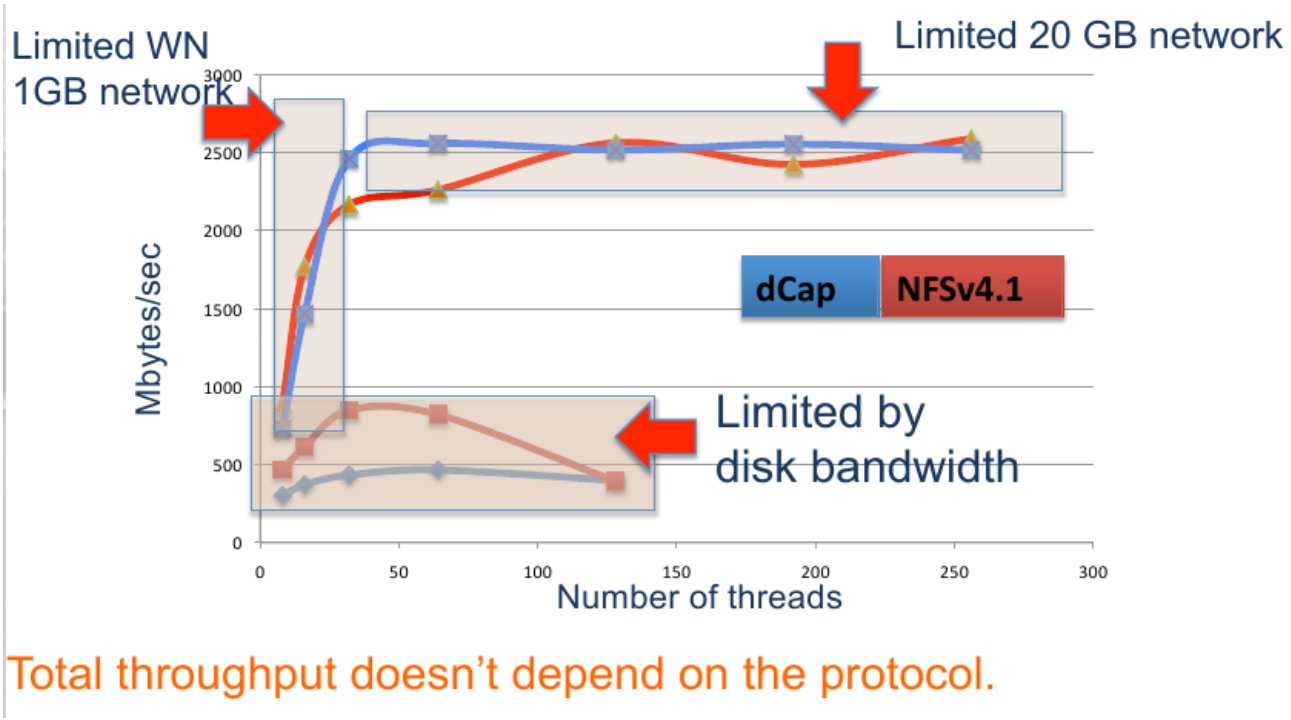


**Abbildung 6:** Testergebnisse im Vergleich mit dCap
©CERN, NFS 4.1 report, Patrick Fuhrmann

## 6 Zusammenfassung

pNFS macht das NFS-Protokoll attraktiv für die Nutzer von HPC-Anwendungen. Dank der Standardisierung und Offenheit des Protokolls und der Schnittstellen erlaubt es einfache Implementierungen, preiswerte Speicherlösungen und Herstellerunabhängigkeit. Die Trennung von Daten und Metadaten ermöglicht, dass Clients direkt und parallel auf die benötigten Daten zugreifen. Innerhalb von Sekunden können mehrere Gigabyte Daten übertragen werden, was die Performance im Vergleich zu vorherigen NFS-Versionen enorm steigert.

## Abkürzungsverzeichnis

| | |
|---|---|
| ACL | Access Control List |
| AFP | Apple Filing Protocol |
| AFS | Andrew Filesystem |
| CERN | European Organisation for Nuclear Research |
| CIFS | Common Internet File System |
| DESY | Deutsches Elektronen Synchrotron |
| FNAL | Fermi National Accelerator Laboratory |
| GPFS | General Parallel File System |
| HPC | High-Performance-Computing |
| IETF | Internet Engineering Task Force |
| LUN | Logical Unit Number |

| | |
|---|---|
| NAS | Networked Attached Storage |
| NDGF | Nordic Data Grid Facility |
| NFS | Network File System |
| OSD | Object-based Storage Device |
| SAN | Storage Area Network |
| SL5 | Scientific Linux 5 |

## Glossar

**NAS**
Fileservices innerhalb eines Netzwerks

**Network File System**
von Sun entwickeltes Protokoll für Netzwerkseicher

## Literatur


[BFK+10] Behrmann, Gerd, Patrick Fuhrmann, Yves Kemp, Tigran Mkrtchyan und Jean-Philippe Baud: *WLCG DM Jamboree : NFS 4.1 - 11 reasons you should care.* Präsentation, 2010. online erhältlich unter `http://www.dcache.org/manuals/2010/20100617-gerd-nfs.pdf`; zuletzt besucht am 1. Juni 2011.

[CER11] CERN: *European Organisation for Nuclear Research.* Webseite, 2011. online erhältlich unter `http://public.web.cern.ch/public/`; zuletzt besucht am 1. Juni 2011.

[dCa10] dCache.org: *dCache Book 1.9.12.* Documentation, 2010. online erhältlich unter `http://www.dcache.org/manuals/Book-1.9.12/`; zuletzt besucht am 1. Juni 2011.

[dCa11] dCache.org: *dCache.org.* Webseite, 2011. online erhältlich unter `http://www.dcache.org`; zuletzt besucht am 1. Juni 2011.

[DES11] DESY: *Deutsches Elektronen-Synchrotron.* Webseite, 2011. online erhältlich unter `http://www.desy.de/`; zuletzt besucht am 1. Juni 2011.

[F+10] Fuhrmann, Pattrick et al.: *Oct GDB : NFS 4.1 Demonstrator Milestone II.* Präsentation, 2010. online erhältlich unter `http://www.dcache.org/manuals/2010/NFS41-demonstrator-milestone-2.pdf`; zuletzt besucht am 1. Juni 2011.

[FB10] Fuhrmann, Patrick und Jean-Philippe Baud: *WLCG London : NFS 4.1 Demonstrator Introduction.* Präsentation, 2010. online erhältlich unter `http://www.dcache.org/manuals/2010/20100707-2-NFS4_demonstrator.pdf`; zuletzt besucht am 1. Juni 2011.

[FBB+11] Fuhrmann, Patrick, Tanja Baranova, Jean-Philippe Baud, Johannes Elmsheuser, Yves Kemp, Maarten Litmaath, Tigran Mkrtchyan, Dmitri Ozerov, Ricardo Rocha, Andrea Sciaba und Hartmut Stadie: *Jan GDB : NFS 4.1 Demonstrator.* Präsentation, 2011. online erhältlich unter `http://www.dcache.org/manuals/2011/20110112-gdb-nfs41.pdf`; zuletzt besucht am 1. Juni 2011.

[FKMO10] Fuhrmann, Patrick, Yves Kemp, Tigran Mkrtchyan und Dmitri Ozerov: *Hepix, Fall 2010 : NFS 4.1 More results.* Präsentation, 2010. online erhältlich unter `http://www.dcache.org/manuals/2010/20101102-hepix-patrick-nfs41.pdf`; zuletzt besucht am 1. Juni 2011.

[FNA11] FNAL: *Fermi National Accelerator Laboratory.* Webseite, 2011. online erhältlich unter `http://www.fnal.gov/`; zuletzt besucht am 1. Juni 2011.

[Fuh10] Fuhrmann, Patrick: *CHEP 2010 : EMI Data, The Introduction.* Präsentation, 2010. online erhältlich unter `http://www.dcache.org/manuals/2010/20101018-chep10-emi-data.pdf`; zuletzt besucht am 1. Juni 2011.

[GSI08] Godson, Garth, Sai Susarla und Rahul Iyer: *Standardizing Storage Clusters.* Webseite, 2008. online erhältlich unter `http://queue.acm.org/detail.cfm?id=1317402`; zuletzt besucht am 1. Juni 2011.

[Int10a] Internet Engineering Task Force: *RFC 5661: nfsv4 minorversion1.* Working document, 2010. online erhältlich unter `http://tools.ietf.org/wg/nfsv4/draft-ietf-nfsv4-minorversion1/`; zuletzt besucht am 1. Juni 2011.

[Int10b] Internet Engineering Task Force: *RFC 5663: nfsv4 pnfs block.* Working document, 2010. online erhältlich unter `http://tools.ietf.org/wg/nfsv4/draft-ietf-nfsv4-pnfs-block/`; zuletzt besucht am 1. Juni 2011.

[Int10c] Internet Engineering Task Force: *RFC 5664: nfsv4 pnfs obj.* Working document, 2010. online erhältlich unter `http://tools.ietf.org/wg/nfsv4/draft-ietf-nfsv4-pnfs-obj/`; zuletzt besucht am 1. Juni 2011.

[K+10] Kemp, Yves et al.: *CHEP 2010 : NFS 4.1 LHC Data Analysis Using NFSv4.1.* Präsentation, 2010. online erhältlich unter `http://www.dcache.org/manuals/2010/CHEP2010-NFS41-kemp.pdf`; zuletzt besucht am 1. Juni 2011.

[Mkr11] Mkrtchyan, Tigran: *pNFS Capabilities in the Real World or pNFS under fire.* Präsentation, 2011. online erhältlich unter `http://www.connectathon.org/talks11/dcache-cthon11.pdf`; zuletzt besucht am 1. Juni 2011.

[NDG11] NDGF: *Nordic Data Grid Facility.* Webseite, 2011. online erhältlich unter `http://www.ndgf.org/`; zuletzt besucht am 1. Juni 2011.

[pNF11] pNFS: *pNFS.* Webseite, 2011. online erhältlich unter `http://www.pnfs.com/`; zuletzt besucht am 1. Juni 2011.

[Str08] Streicher, Martin: *Scale your file system with Parallel NFS.* Webseite, 2008. online erhältlich unter `http://www.ibm.com/developerworks/linux/library/l-pnfs/`; zuletzt besucht am 1. Juni 2011.