NFSv4.1 — Using pNFS

SNIA™ WEBCAST

Presented by:
Alex McDonald
CTO Office, NetApp

HOSTED BY THE
ETHERNET STORAGE FORUM

Webcast Presenter

Alex McDonald
Office of the CTO
NetApp

Alex McDonald joined NetApp in 2005, after more than 30 years in a variety of roles with some of the best known names in the software industry .

With a background in software development, support, sales and a period as an independent consultant, Alex is now part of NetApp's Office of the CTO that supports industry activities and promotes technology & standards based solutions, and is co-chair of the SNIA Cloud Storage Initiative, and co-chair of the SNIA File Protocols Special Interest Group.

SNIA Europe™

Ethernet Storage Forum Members

The SNIA Ethernet Storage Forum (ESF) focuses on educating end-users about Ethernet-connected storage networking technologies.

SNIA Europe™

SNIA's NFS Special Interest Group

- File Protocol SIG drives adoption and understanding of SMB and NFS across vendors to constituents
  - Marketing, industry adoption, Open Source updates
- NetApp, EMC, Panasas and Sun founders
- White papers on migration from NFSv3 to NFSv4
  - An Overview of NFSv4; NFSv4, NFSv4.1, pNFS, and proposed NFSv4.2 features
  - Migrating from NFSv3 to NFSv4

SNIA Ethernet Storage Forum

Learn more about us at: www.snia.org/forums/esf

SNIA Europe™

Previous SNIA NFS Presentations

- BrightTalk SNIA Channel NFS Mini Series
- NFSv4.1, pNFS & FedFS Protocol Development
  - Part1 – Four Reasons for NFSv4
    - Discusses the reasons behind the development of NFSv4 and beyond, and the need for a better-than-NFSv3 protocol
  - Part2 – Advances in NFS – NFSv4.1 and pNFS
    - An overview and some details on NFSv4.1, pNFS (parallel NFS), and FedFS (the Federated filesystem); and a high level overview of proposed NFSv4.2 features
  - Part3 – Planning for a Smooth Migration
    - The key issues to consider when migrating from NFSv3 or implementing new applications with NFSv4.1; Unicode, security with Kerberos, statefulness, selecting the application and other aspects.
- Slides available from
  - http://snia.org/forums/esf/knowledge/webcasts

BrightTALK™

SNIA Europe™

The Four Reasons for NFSv4.1

	Functional	Business Benefit
Security	ACLs for authorization Kerberos for authentication	Compliance, improved access, storage efficiency, WAN use
High availability	Client and server lease management with fail over	High Availability, Operations simplicity, cost containment
Single namespace	Pseudo directory system, FedFS	Reduction in administration & management
Performance	Multiple read, write, delete operations per RPC call Delegate locks, read and write procedures to clients Parallelised I/O	Better network utilization for all NFS clients Leverage NFS client hardware for better I/O

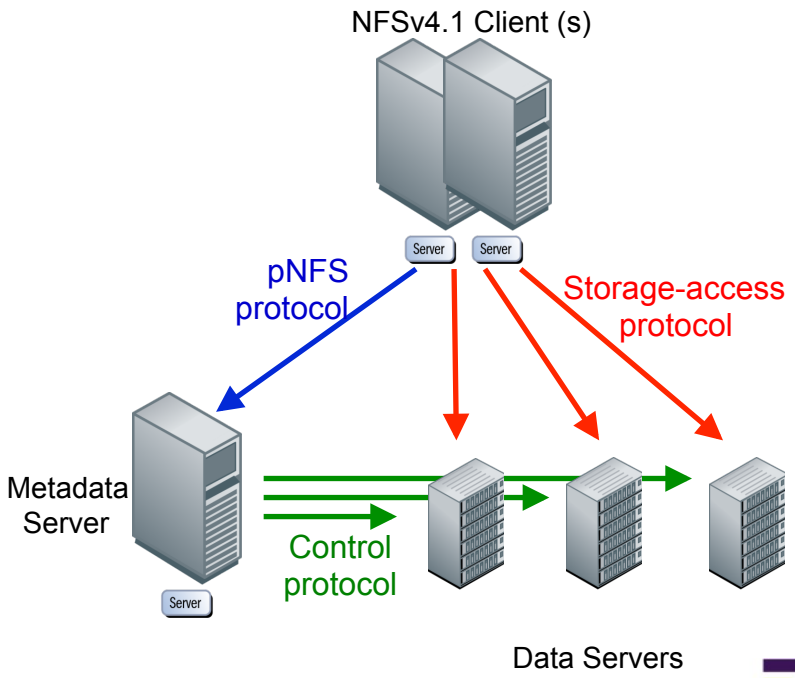
Agenda

- We'll cover
  - Overview of pNFS terminology and operation
  - How clients & servers co-operate to provide parallelism while supporting data consistency
  - Some implementation considerations
- This is a high level overview
  - But more technical content for background
  - Use SNIA white papers and vendors (both client & server) to help you implement

You've Done NFSv4.1; now for pNFS

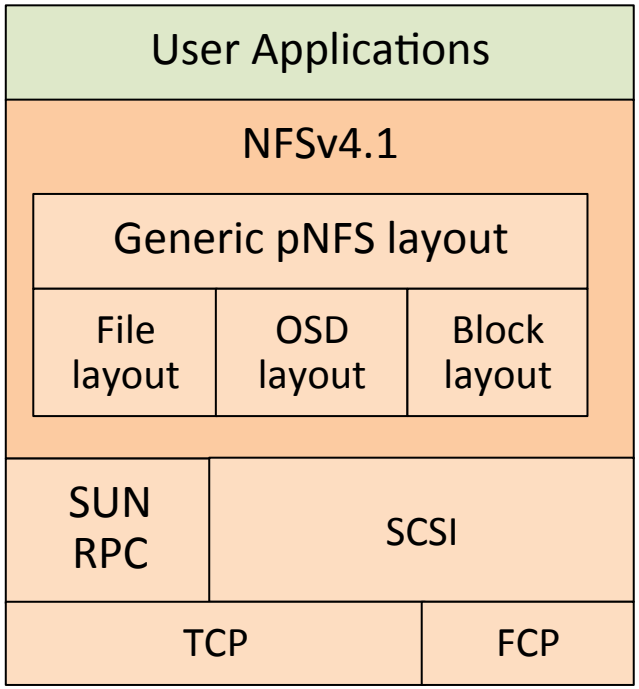
- NFSv4.1 (pNFS) can aggregate bandwidth
  - Modern approach; relieves issues associated with point-to-point connections

- pNFS Client
  - Client read/write a file
  - Server grants permission
  - File layout (stripe map) is given to the client
  - Client parallel R/W directly to data servers
- Removes IO Bottlenecks
  - No single storage node is a bottleneck
  - Improves large file performance
- Improves Management
  - Data and clients are load balanced
  - Single Namespace



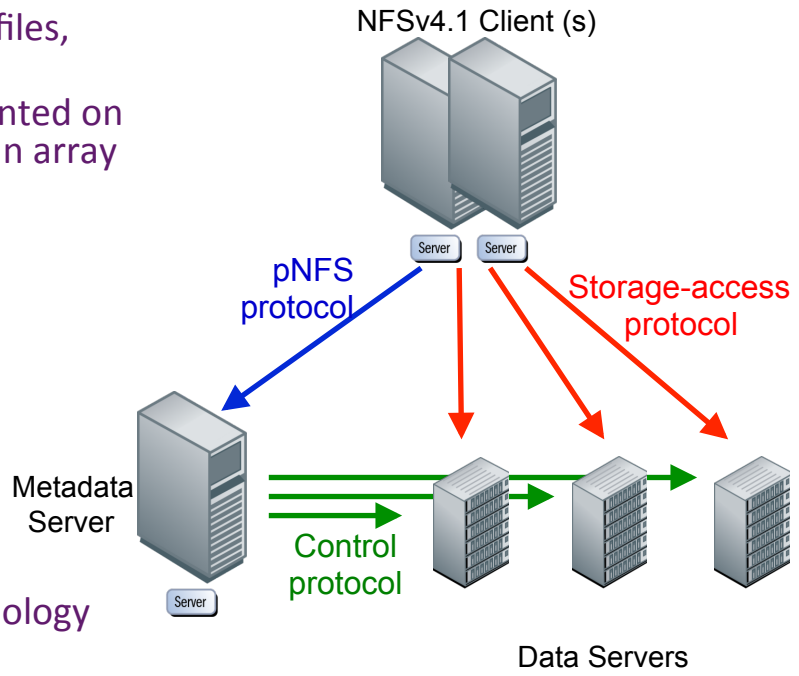
Relationship of pNFS to NFSv4.1

- RFC 3530bis – Network File System (NFS) Version 4 Protocol
  - NFSv4 (updated from RFC 3530 based on experience)
- RFC 5661 – Network File System (NFS) Version 4 Minor Version 1 Protocol
  - Specifies Sessions, Directory Delegations, and parallel NFS (pNFS) for files
- RFC 5663 - Parallel NFS (pNFS) Block/ Volume Layout
- RFC 5664 - Object-Based Parallel NFS (pNFS) Operations
- pNFS is dependant on session support, which is only available in NFSv4.1



pNFS Terminology

- Metadata Server; the MDS
  - Maintains information about location and layout of files, objects or block data on data servers
  - Shown as a separate entity, but commonly implemented on one or across more than one data server as part of an array
- pNFS protocol
  - Extended protocol over NFSv4.1
  - Client to MDS communication
- Storage access protocol
  - Files; NFS operations
  - Objects: OSD SCSI objects protocol (OSD2)
  - Blocks; SCSI blocks (iSCSI, FCP)
- Control protocol
  - Not standardised; each vendor uses their own technology to do this
- Layout
  - Description of devices and sector maps for the data stored on the data servers
  - 3 types; files, block and object
- Callback
  - Asynchronous RPC calls used to control the behavior of the client during pNFS operations



pNFS Operations

- Client requests layout from MDS
- Layout maps the file/object/ block to data server addresses and locations
- Client uses layout to perform direct I/O to the storage layer
- MDS or data server can recall the layout at any time using callbacks
- Client commits changes and releases the layout when complete
- pNFS is optional
  - Client can fall back to NFSv4

- pNFS operations
  - LAYOUTCOMMIT Servers commit the layout and update the meta-data maps
  - LAYOUTRETURN Returns the layout or the new layout, if the data is modified
  - GETDEVICEINFO Client gets updated information on a data server in the storage cluster
  - GETDEVICELIST Clients requests the list of all data servers participating in the storage cluster
  - CB_LAYOUT Server recalls the data layout from a client if conflicts are detected

pNFS Pre-requisites

- NFSv4.1 and pNFS capable server
  - Contact your NAS vendor for availability
  - Commercial products available for all of files, blocks and object types
  - Open source Linux pNFS server in development
    - http://wiki.linux-nfs.org/wiki/index.php/PNFS_Development
- pNFS capable client
  - Linux to date
  - See previous BrightTalks
    - Part3 – Planning for a Smooth Migration

Linux Client and NFSv4.1

- Upstream (Linus) Linux NFSv4.1 client support
  - Basic client in Kernel 2.6.32
  - pNFS support (files layout type) in Kernel 2.6.39
  - Support for the 'objects' and 'blocks' layouts was merged in Kernel 3.0 and 3.1 respectively
- Full read and write support for all three layout types in the upstream kernel
  - Blocks, files and objects
  - O_DIRECT reads and writes supported

Linux Client and NFSv4.1

- pNFS client support in distributions
  - Fedora 15 was first for pNFS files
  - Kernel 2.6.40 (released August 2011)
- Red Hat Enterprise Linux (RHEL)
  - “Technical preview" support for NFSv4.1 and for the pNFS files layout type in version 6.2, 6.3
  - Full support in RHEL6.4, announced Feb 2013
- Ubuntu, SUSE & other distributions
  - Possible to upgrade to NFSv4.1
- No support in Solaris
  - Both server and client are NFSv4 only

pNFS Files Mount

- RHEL6.4 pNFS mount
  - mount -o minorversion=1 server:/filesystem /mnt
- Check
  - (output edited)

```
/proc/self/mountstats

device 172.16.92.172:/filesystem mounted on /mnt with fstype
nfs4 statvers=1.1
opts: …,vers=4.1, …
nfsv4: …,sessions,pnfs=nfs_layout_nfsv41_files
…
```

pNFS Client Mount


```
    47 27.086618  172.17.40.185      172.17.40.171      NFS      282 V4 Call (Reply In 48) EXCHANGE_ID
    48 27.086762  172.17.40.171      172.17.40.185      NFS      266 V4 Reply (Call In 47) EXCHANGE_ID
    49 27.086883  172.17.40.185      172.17.40.171      NFS      330 V4 Call (Reply In 51) CREATE_SESSION
    50 27.087003  172.17.40.171      172.17.40.185      NFS      146 V1 CB_NULL Call (Reply In 53)
    51 27.087032  172.17.40.171      172.17.40.185      NFS      194 V4 Reply (Call In 49) CREATE_SESSION
⊞ Ethernet II, Src: Netapp_20:7a:42 (00:a0:98:20:7a:42), Dst: IntelCor_2b:40:06 (00:1b:21:2b:40:06)
⊞ Internet Protocol Version 4, Src: 172.17.40.171 (172.17.40.171), Dst: 172.17.40.185 (172.17.40.185)
⊞ Transmission Control Protocol, Src Port: nfs (2049), Dst Port: 1007 (1007), Seq: 29, Ack: 261, Len: 200
⊞ Remote Procedure Call, Type:Reply XID:0x634bd45a
⊟ Network File System, Ops(1): EXCHANGE_ID
    [Program Version: 4]
    [V4 Procedure: COMPOUND (1)]
    Status: NFS4_OK (0)
  ⊞ Tag: <EMPTY>
  ⊟ Operations (count: 1)
    ⊟ Opcode: EXCHANGE_ID (42)
        Status: NFS4_OK (0)
        clientid: 0x6387220000000004
        seqid: 0x00000001
      ⊟ eir_flags: 0x00060100
          0... .... .... .... .... .... .... .... = EXCHGID4_FLAG_CONFIRMED_R: Not set
          .0.. .... .... .... .... .... .... .... = EXCHGID4_FLAG_UPD_CONFIRMED_REC_A: Not set
          .... .... .... .1.. .... .... .... .... = EXCHGID4_FLAG_USE_PNFS_DS: Set
          .... .... .... ..1. .... .... .... .... = EXCHGID4_FLAG_USE_PNFS_MDS: Set
          .... .... .... ...0 .... .... .... .... = EXCHGID4_FLAG_USE_NON_PNFS: Not set
          .... .... .... .... .... ...1 .... .... = EXCHGID4_FLAG_BIND_PRINC_STATEID: Set
          .... .... .... .... .... .... .... ..0. = EXCHGID4_FLAG_SUPP_MOVED_MIGR: Not set
          .... .... .... .... .... .... .... ...0 = EXCHGID4_FLAG_SUPP_MOVED_REFER: Not set
        eia_state_protect: SP4_NONE (0)
```

172.17.40.185 – IP address of the pNFS client
172.17.40.171 – IP address of the server

Client and Server handshake to determine respective Capabilities. The Cluster replies with MDS and DS flags set, indicating capability for both

pNFS Client to MDS


```
117 44.370851  172.17.40.185   172.17.40.171   NFS   418 V4 Call (Reply In 118) OPEN DH:0x7f69f7d7/testfile5
118 44.470682  172.17.40.171   172.17.40.185   NFS   566 V4 Reply (Call In 117) OPEN StateID:0xa36e
119 44.470856  172.17.40.185   172.17.40.171   NFS   338 V4 Call (Reply In 120) SETATTR FH:0x4c99adea
120 44.471391  172.17.40.171   172.17.40.185   NFS   318 V4 Reply (Call In 119) SETATTR
121 44.477141  172.17.40.185   172.17.40.171   NFS   342 V4 Call (Reply In 122) LAYOUTGET
122 44.477244  172.17.40.171   172.17.40.185   NFS   306 V4 Reply (Call In 121) LAYOUTGET
123 44.477406  172.17.40.185   172.17.40.171   NFS   274 V4 Call (Reply In 124) GETDEVINFO
124 44.477501  172.17.40.171   172.17.40.185   NFS   218 V4 Reply (Call In 123) GETDEVINFO
129 44.477982  172.17.40.185   172.17.40.173   NFS   110 V4 NULL Call (Reply In 130)
130 44.478154  172.17.40.173   172.17.40.185   NFS    94 V4 NULL Reply (Call In 129)
```

```
    Status: NFS4_OK (0)
    sessionid: 0000000463872200000000000000000
    seqid: 0x00000017
    slot ID: 0
    high slot id: 0
    target high slot id: 15
    status: 0
⊟ Opcode: GETDEVINFO (47)
    Status: NFS4_OK (0)
    layout type: LAYOUT4_NFSV4_1_FILES (1)
    device index: 0
  ⊟ r_netid: tcp
      length: 3
      contents: tcp
      fill bytes: opaque data
  ⊟ r_addr: 172.17.40.173.8.1
      length: 17
      contents: 172.17.40.173.8.1
      fill bytes: opaque data
    [Main Opcode: GETDEVINFO (47)]
```

The OPEN and SETATTR are sent to the MDS

MDS LAYOUT to pNFS Client


```
121 44.477141  172.17.40.185     172.17.40.171     NFS     342 V4 Call (Reply In 122) LAYOUTGET
122 44.477244  172.17.40.171     172.17.40.185     NFS     306 V4 Reply (Call In 121) LAYOUTGET
123 44.477406  172.17.40.185     172.17.40.171     NFS     274 V4 Call (Reply In 124) GETDEVINFO
124 44.477501  172.17.40.171     172.17.40.185     NFS     218 V4 Reply (Call In 123) GETDEVINFO
129 44.477982  172.17.40.185     172.17.40.173     NFS     110 V4 NULL Call (Reply In 130)
130 44.478154  172.17.40.173     172.17.40.185     NFS      94 V4 NULL Reply (Call In 129)
```

```
    Status: NFS4_OK (0)
  ⊟ Opcode: LAYOUTGET (50)
    Status: NFS4_OK (0)
    return on close?: No
  ⊟ stateid
      [StateID Hash: 0x28fd]
      seqid: 0x00000001
      Data: 032287634f000e0000000000
  ⊟ Layout Segment (count: 1)
      offset: 0
      length: 18446744073709551615
      IO mode: IOMODE_RW (2)
      layout type: LAYOUT4_NFSV4_1_FILES (1)
      device ID: 0101010016040080000000001000000
      nfl_util: 0x00010000
      first stripe to use index: 0
      offset: 0
    ⊞ File Handles (count: 1)
  [Main Opcode: LAYOUTGET (50)]
```

Before reading or writing data, the pNFS client requests the layout

The map of data servers and file handles is returned

pNFS Client DEVICEINFO from MDS


```
117 44.370851  172.17.40.185   172.17.40.171   NFS   418 V4 Call (Reply In 118) OPEN DH:0x7f69f7d7/testfile5
118 44.470682  172.17.40.171   172.17.40.185   NFS   566 V4 Reply (Call In 117) OPEN StateID:0xa36e
119 44.470856  172.17.40.185   172.17.40.171   NFS   338 V4 Call (Reply In 120) SETATTR FH:0x4c99adea
120 44.471391  172.17.40.171   172.17.40.185   NFS   318 V4 Reply (Call In 119) SETATTR
121 44.477141  172.17.40.185   172.17.40.171   NFS   342 V4 Call (Reply In 122) LAYOUTGET
122 44.477244  172.17.40.171   172.17.40.185   NFS   306 V4 Reply (Call In 121) LAYOUTGET
123 44.477406  172.17.40.185   172.17.40.171   NFS   274 V4 Call (Reply In 124) GETDEVINFO
124 44.477501  172.17.40.171   172.17.40.185   NFS   218 V4 Reply (Call In 123) GETDEVINFO
129 44.477982  172.17.40.185   172.17.40.173   NFS   110 V4 NULL Call (Reply In 130)
130 44.478154  172.17.40.173   172.17.40.185   NFS    94 V4 NULL Reply (Call In 129)
```

```
    Status: NFS4_OK (0)
    sessionid: 0000000463872200000000000000000000
    seqid: 0x00000017
    slot ID: 0
    high slot id: 0
    target high slot id: 15
    status: 0
  ⊟ Opcode: GETDEVINFO (47)
    Status: NFS4_OK (0)
    layout type: LAYOUT4_NFSV4_1_FILES (1)
    device index: 0
  ⊟ r_netid: tcp
      length: 3
      contents: tcp
      fill bytes: opaque data
  ⊟ r_addr: 172.17.40.173.8.1
      length: 17
      contents: 172.17.40.173.8.1
      fill bytes: opaque data
    [Main Opcode: GETDEVINFO (47)]
```

Meta-data node provides the pNFS client with the IP information for the DS. In this example – 172.17.40.173

Information is cached for life of the layout or until recalled (for example, when the data is moved)

pNFS Client Uses Direct Data Path


```
123 44.477406  172.17.40.185   172.17.40.171   NFS   274 V4 Call (Reply In 124) GETDEVINFO
124 44.477501  172.17.40.171   172.17.40.185   NFS   218 V4 Reply (Call In 123) GETDEVINFO
129 44.477982  172.17.40.185   172.17.40.173   NFS   110 V4 NULL Call (Reply In 130)
130 44.478154  172.17.40.173   172.17.40.185   NFS    94 V4 NULL Reply (Call In 129)
132 44.478663  172.17.40.185   172.17.40.173   NFS   282 V4 Call (Reply In 133) EXCHANGE_ID
133 44.478784  172.17.40.173   172.17.40.185   NFS   266 V4 Reply (Call In 132) EXCHANGE_ID
134 44.478918  172.17.40.185   172.17.40.173   NFS   330 V4 Call CREATE_SESSION
163 60.480000  172.17.40.185   172.17.40.173   NFS   330 V4 Call (Reply In 206) CREATE_SESSION
169 64.476795  172.17.40.185   172.17.40.171   NFS   242 V4 Call (Reply In 170) SEQUENCE
170 64.476916  172.17.40.171   172.17.40.185   NFS   150 V4 Reply (Call In 169) SEQUENCE
191 76.480717  172.17.40.185   172.17.40.173   NFS   330 V4 Call CREATE_SESSION
```

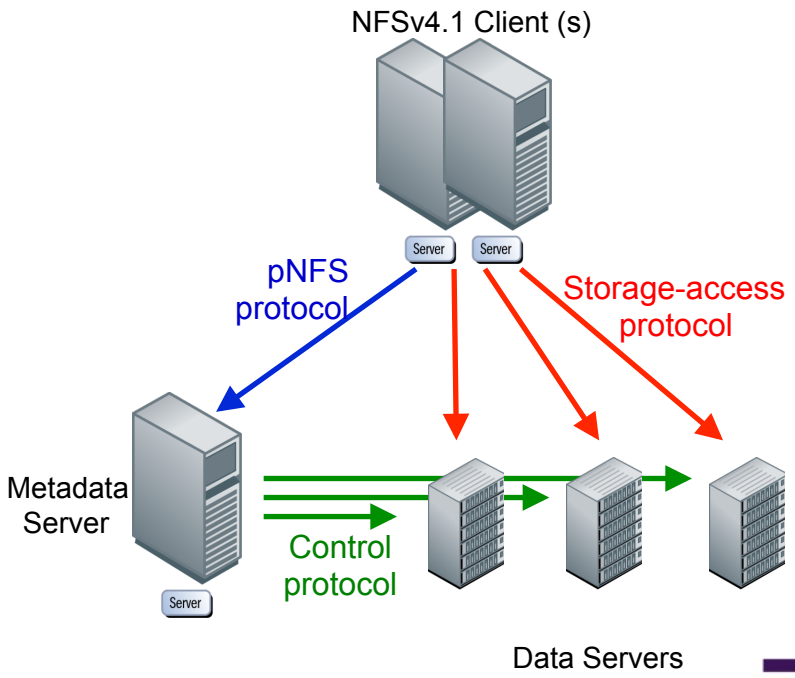
```
⊟ Network File System, Ops(2): SEQUENCE GETDEVINFO
    [Program Version: 4]
    [V4 Procedure: COMPOUND (1)]
    Status: NFS4_OK (0)
  ⊞ Tag: <EMPTY>
  ⊟ Operations (count: 2)
    ⊞ Opcode: SEQUENCE (53)
    ⊟ Opcode: GETDEVINFO (47)
        Status: NFS4_OK (0)
        layout type: LAYOUT4_NFSV4_1_FILES (1)
        device index: 0
      ⊟ r_netid: tcp
          length: 3
          contents: tcp
          fill bytes: opaque data
      ⊟ r_addr: 172.17.40.173.8.1
          length: 17
          contents: 172.17.40.173.8.1
          fill bytes: opaque data
      [Main Opcode: GETDEVINFO (47)]
```

Now the pNFS client is reaching out to the remote volume on a direct path using IP address 172.17.40.173.

In Summary: The Benefits of pNFS

- NFSv4.1 (pNFS) can aggregate bandwidth
  - Modern approach; relieves issues associated with point-to-point connections

- pNFS Client
  - Client read/write a file
  - Server grants permission
  - File layout (stripe map) is given to the client
  - Client parallel R/W directly to data servers
- Removes IO Bottlenecks
  - No single storage node is a bottleneck
  - Improves large file performance
- Improves Management
  - Data and clients are load balanced
  - Single Namespace



Other NFS Performance Capabilities

- Trunking (NFSv4.1 & pNFS)
  - A single data server connection limits data throughput based on protocol
  - Trunking "bundles" connections into a single pipe
    - Open multiple sessions via different physical Ethernet connections to the same file handle/data server resource
  - Expands throughput and can reduce latency
  - No implementations as yet
- Compound operations (NFSv4 and above)
  - Example: LOOKUP, OPEN, READ, CLOSE as a single RPC call
  - Benefits WAN operations
- Caching & delegation (NFSv4 and NFSv4.1)
  - Allows client and server to agree on data that will be processed by the client
  - Reduces IO and provides data locality

Summary/Call to Action

- Start using NFSv4.1 today
  - NFSv4.2 nearing approval
  - pNFS offers performance support for modern NAS devices
- Planning is key
  - Application, issues & actions to ensure smooth implementations
- pNFS
  - First open standard for parallel I/O across the network
  - Ask vendors to include NFSv4.1 and pNFS support for client/ servers
  - pNFS has wide industry support
  - Commercial implementations and open source

Question & Answer

To download this Webcast

after the presentation, go to

http://www.snia.org/about/socialmedia/