NFSv4 and Petascale Data Management

Andy Adamson

CITI, University of Michigan

andros@umich.edu

Outline

- GridNFS
  - Integrates NFSv4 into the ecology of Grid middleware
- Linux pNFS prototype
  - NFSv4.1 minor version extension for direct parallel data access

GridNFS - Joining NFSv4 and Grid Middleware

- Globus GSI support
  - Grid X.509 trust infrastructure
- Fine-grained access control with foreign user and foreign group support
- High performance secure file system access for jobs scheduled in an indeterminate future.
- Flexible file name space construction and management

GridNFS - Strong Security

- Security is mandatory for NFSv4
  - RPCSEC_GSS Data privacy, integrity, mutual authentication
  - Kerberos v5, SPKM-3/LIPKEY mandatory security mechanisms (RFC 3530)
- SPKM-3: X.509 GSS mechanism (RFC 2847)
  - Anonymous secure channel for LIPKEY username/password
  - User X.509 credentials can also be used.
  - Plays well with Grid X.509 trust infrastructure
- Linux and Hummingbird (Windows) SPKM-3 implementations

NFSv4 and X.509 Based Security

- PKINIT
  - Public Key based initial authentication in Kerberos
    - X.509 credential, not password.
  - Implemented by CITI for MIT Kerberos
- SPKM-3 based on SPKM-1 and SPKM-2
  - RFC 2847 required work to pass IETF
    - old algorithms, under specified fields, old style error tokens…
  - Rejected by the IETF Security Area working group process
- PKU2U
  - “PKINIT without a KDC”
    - No Kerberos Infrastructure required.
    - Anonymous and X.509 credential support
  - Proposed X.509 based GSS mechanism

GridNFS - Data Access Control

- NFSv4 *requires* ACLs
  - Similar to Windows ACLs
  - Superset of POSIX ACLs
- “On the wire” names are Unicode strings
  - *Not* integers (except in AUTH_SYS RPC)…
  - Allows ACLs for foreign users
- *GridNFS adds secure mechanisms to map foreign users to local identity*

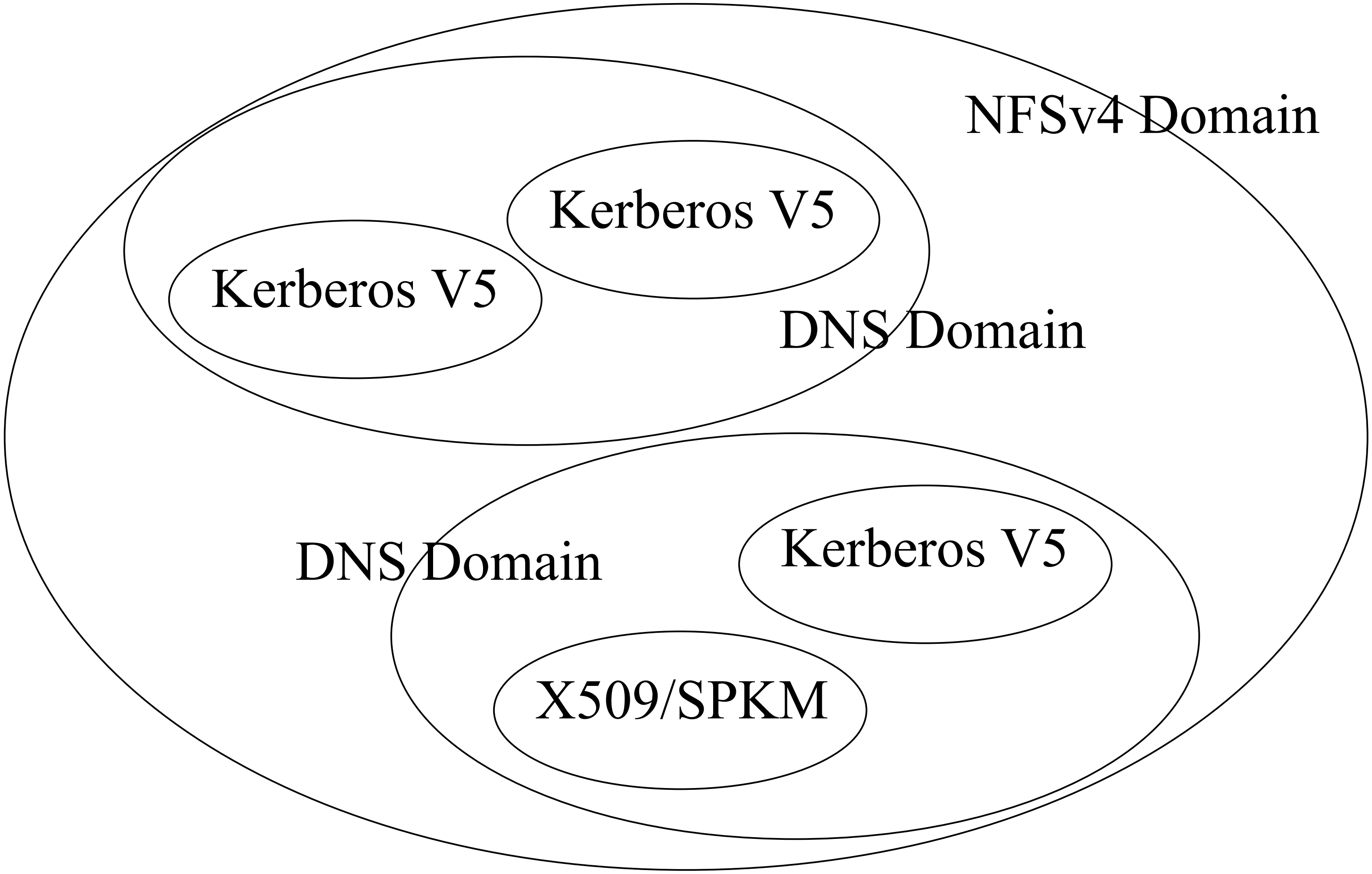
GridNFS - Data Access Control

- NFSv4 places two types of names on the wire.
- RPCSEC_GSS per security mechanism name in the RPC credential
  - Used for identity and mapped to a UID on the client and server
  - Kerberos V: joe@ANYTHING.ORG
  - X.509: OU=US, OU=State, OU=Arbitrary Inc, CN=joe
- NFSv4 domain name in GETATTR and SETATTR operations
  - ACL name form for users and groups
  - Joe@arbitrary.domain.org

GridNFS - Data Access Control

- NFSv4 domain = unique UID/GID to name mapping
  - Administrative unit for NFSv4
  - AFS has a Cell, NFSv4 has a domain
- Multiple security authorities possible (GSS Names)
  - Multiple Kerberos realms and/or PKI Certificate Authorities result in multiple RPCSEC_GSS credential names
    - Kerberos v5 principal
    - SPKM-3 X.509 DN
- Multiple DNS - NIS domains possible (ACL Name)
  - Pick one to DNS domain to be the NFSv4 domain name
  - NFSv4 GETATTR/SETATTR ACL name
    - user@nfsv4domain

NFSv4 Domain



New LDAP Attributes

- We created a new LDAP object to hold two new LDAP attributes for NFSv4 ID mapping
  - GSSAuthName
  - NFSv4Name
- We associate *one* NFS4Name attribute with an RFC2307 NSS-LDAP posixAccount to hold the users v4 domain name
  - Joe@arbitrary.domain.org
- We associate *multiple* GSSAuthNames with the posixAccount
  - Kerberos V: joe@ANYTHING.ORG
  - PKI: OU=US, OU=State, OU=Arbitrary Inc, CN=joe

GridNFS - Foreign Users

- Mapping of NFSv4 and GSS principal names to UID/GIDs allows for mapping of foreign user and group names.
  - Start with a cross domain security relationship
    - Kerberos cross realm, PKI CA certification path, etc
  - Assign a local UID (posixAccount) and an NFS4Name to a foreign user
  - Associate their GSS principal(s) with the local UID via GSSAuthName.
  - Tested at Sandia National Labs
- NFSv4 provides GridNFS with flexible trust virtualization.

NFSv4 Pseudo File System

- Unlike NFSv2/3 which exports volumes
- NFSv4 exports a virtual server root and read-only pseudo file system
    - onto which exported volumes are mounted as pseudo file system leaves
    - giving server control over the name space visible to the client
- Clients mount ‘/’ and users traverse the pseudo file system to reach exported volumes.

NFSv4 Pseudo File System

- ‘Normal’ Pseudo file system leaves are mount points for volumes with data
- Pseudo file system leaves can also have no data, but are re-direction points called *referrals*
  - Used for global namespace construction and so never had data mounted
  - Left over from a file system migration

File System Redirection

- Client tries to access a referral
- Receives special error (“moved”)
- Retrieves *fs_locations* attribute (GETATTR)
  - Value is list of {server, path} pairs giving location(s) of the file system
- Client selects one
  - Policy is effected by server in constructing the list
  - and by client in making selection

NFSv4 Global Name Space

- NFSv4 allows the construction of a global namespace for data management
  - An AFS-like global or VO project namespace
- Client mounts /nfs/<NFSv4 domain>
  - /nfs: on client file system
  - <NFSv4 domain>: DNS SRV specifies server to mount which exports the root of the NFSv4 domain namespace.
- User traverses /nfs/<NFSv4 domain>/foo/bar
  - /foo/bar are served by NFSv4 *Pseudo File System* and *fs_locations* redirection.

GridNFS and OSG

- Objective: Back OSG cluster with NFSv4, and run a job in the future with secure NFSv4 file access.
  - Installed Condor and OSG into Linux NFSv4 SPKM-3 only exported file systems.
    - Added LDAP entries and X.509 certificates for OSG daemons
  - Provide a MyProxy server to store job-long renewable proxy credentials
    - Place user long-term GSI X.509 credentials in MyProxy
  - Use our MGRID portal as a job scheduling interface.

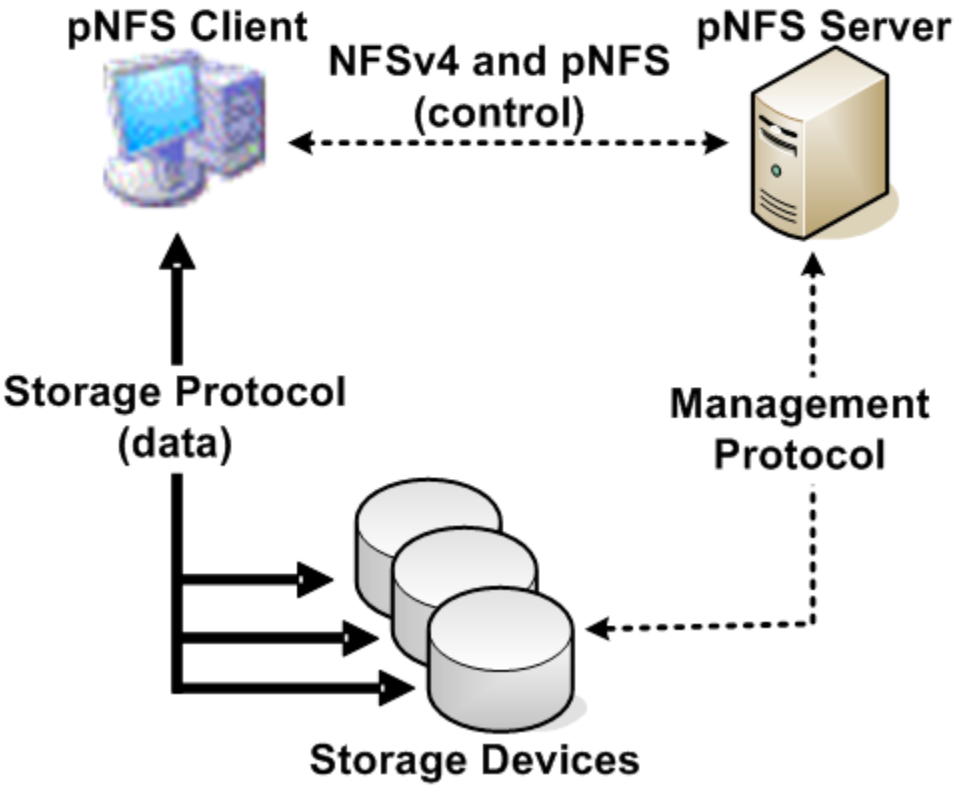
GridNFS and Condor

- Submit a job through the portal to execute in the future
  - Portal propagates proxy credentials to an execute node
  - All jobs use Condor’s USER_JOB_WRAPPER
    - Proxy credentials available to NFSv4
    - Renews about-to-expire credentials via MyProxy
- Condor Integration - It’s all about credentials
  - Proxy credential propagation is tied into File I/O mechanism
  - Setup of a job’s stderr/stdout is done by the daemon process
    - Lacks necessary credentials.

WAN Performance

- NFS4.0 WAN performance superior to NFSv3, AFS
  - File delegations
- NFSv4.1 pNFS removes single client to single server bottleneck
  - Divides NFSv4 into control and data paths
  - Client can stripe I/O across multiple storage devices
  - Common client for previously proprietary file systems
- *Parallel NFS provides high performance secure file system access*

pNFS Architecture



Parallel NFS

- pNFS Client storage protocols
  - File (NFSv4.1)
  - Object
  - Block (SCSI)
  - Other
- Server management protocol not specified
  - Correct NFSv4 behaviour is specified.
- Normal I/O through pNFS server always available
  - Used for low bandwidth requests, failure recovery, etc.

Parallel NFS

- Linux pNFS generic client and server implementation
  - API for opaque storage type specific payloads
  - Client: API to storage type module
  - Server: API to exported pNFS file system
    - File system does most of the work
- NFSv4.1 minor version draft under intense review
  - Sessions (required) and pNFS interoperability tested at Austin Bakeathon, June 11-15th
- Cross storage protocol data transfer accomplished
  - Linux pNFS client copied data between Object and File data stores.

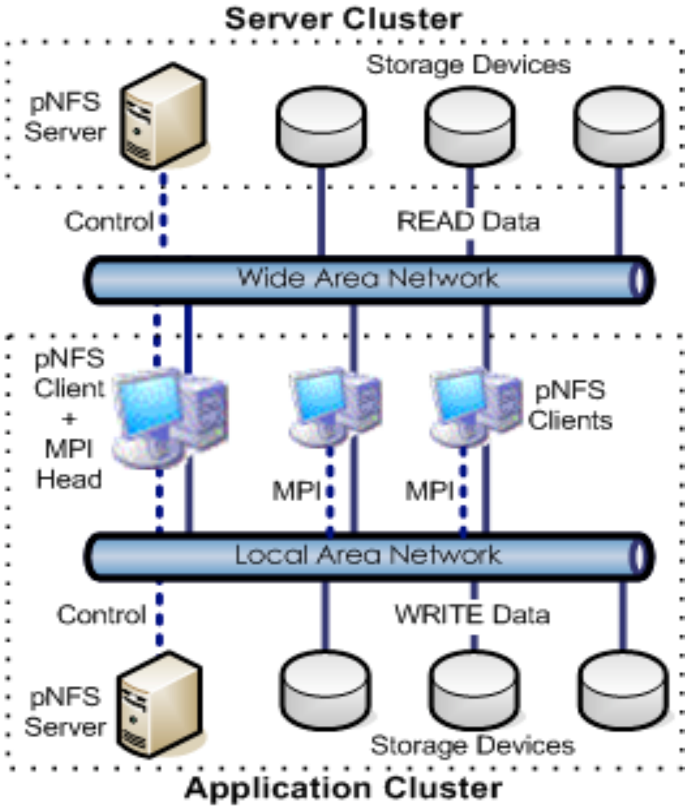
WAN Performance “Smoke” Testing

- UMICH campus 10Gbit net
  - Part of the Ultralight network
- pNFS client with 10Gbit NIC
- 8 node GPFS pNFS server cluster with 1Gbit NICs
  - FC attached storage - 3 to 4 year old equipment
- Both run Linux pNFS file storage protocol (NFSv4.1)
- iozone tests with increasing stripe size
  - From 1 to 5 storage servers
  - No tuning.
- Read and write throughput increased linearly
  - Read: 0.75 to 4.0 Gbits/sec
  - Write: 0.43 to 1.2 Gbits/sec

Current pNFS WAN Testing

- Repeating tests across UMICH 10Gbit network
- Run same tests from CERN
  - Ultralight network
  - Mirrored client hardware at UMICH and CERN
- Tuning client and server
- Comparative tests with GridFTP
- MPI Cluster to cluster testing
  - Copy data from clusters across the WAN
  - Clusters using different storage protocols are of interest
- Looking for partners

MPI and pNFS Exported Clusters



pNFS: Embraced by Industry!

- NFSv4.1 storage protocol
  - Linux and Solaris Client
  - Solaris, IBM GPFS, dCache, Linux Server
- Object storage protocol
  - Linux client
  - Linux server exporting Panasas pNFS file system
- Block storage protocol
  - Linux client
  - EMC server exporting pNFS file system
- Other pNFS implementations underway
  - I’m sure this is an incomplete list…..

Conclusion

- GridNFS and pNFS combine and integrate standard Internet protocols, promising compatibility with standards-compliant desktop and enterprise network services.

Questions?

http://www.citi.umich.edu