Parallel NFS

Dr. Oliver Tennert, Head of Technology

transtec

sambaXP 2008
Göttingen, 18.04.2008

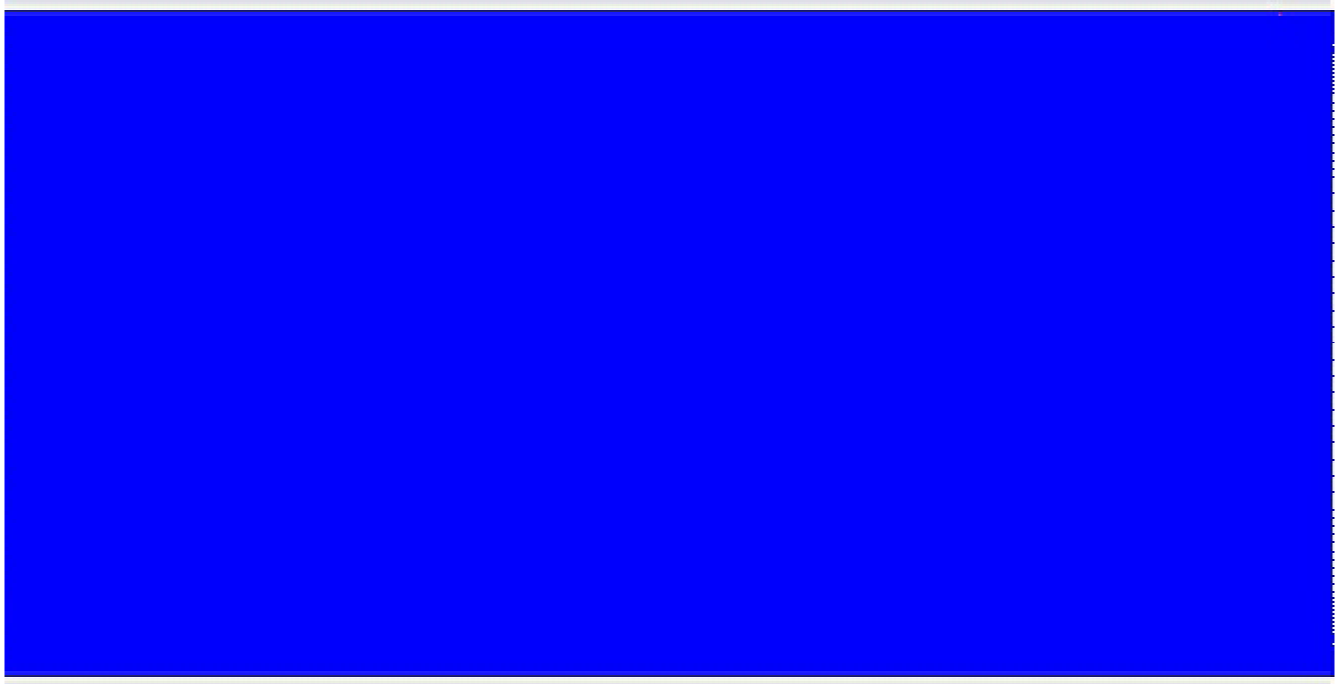
Overview

- Motivation: Why Parallel NFS?
- What is pNFS?
- How does it work?
- Some numbers…

Siberia 30.06.1908

- massive explosion in Tunguska Region, Central Siberia
- 2,150 km^2 devastated
- 60-80 million trees felled within seconds
- est. 5,0 earthquake from the blast
- most probable explanation:
  - crash of **massive meteorite**
  - physical mass: about **10,000 tons**
  - detonation in **10 km altitude**
  - at a speed of about **70,000 km/h**
  - equivalent of **10-15 megatons TNT**

High Performance Computing



Source: www.sandia.gov

Storage Demands in HPC

- need for **computing power**
  - due to need to run larger and more accurate models
  - more CPUs, more cores, more nodes, more RAM
- need for **network performance**
  - more highly paralellized jobs
  - high-speed interconnects (10GbE, InfiniBand,...)

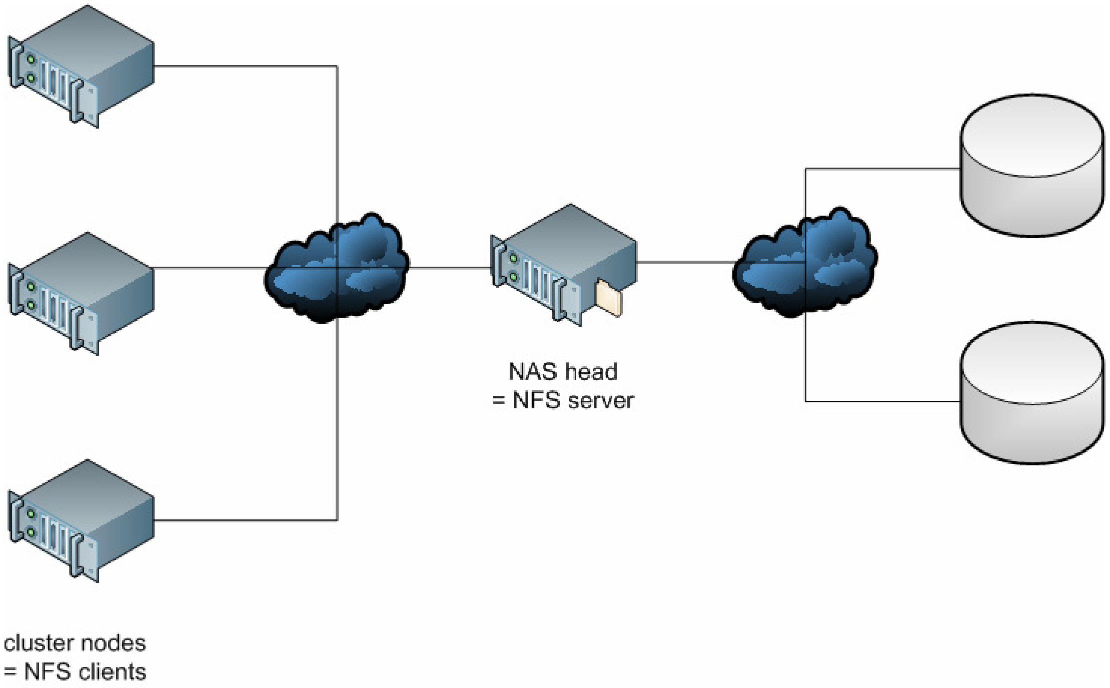
→ **massive explosion of data sets**

→ demand for

- **large storage capacity**
- **high bandwidth**
- **low latency**

Today: NFS

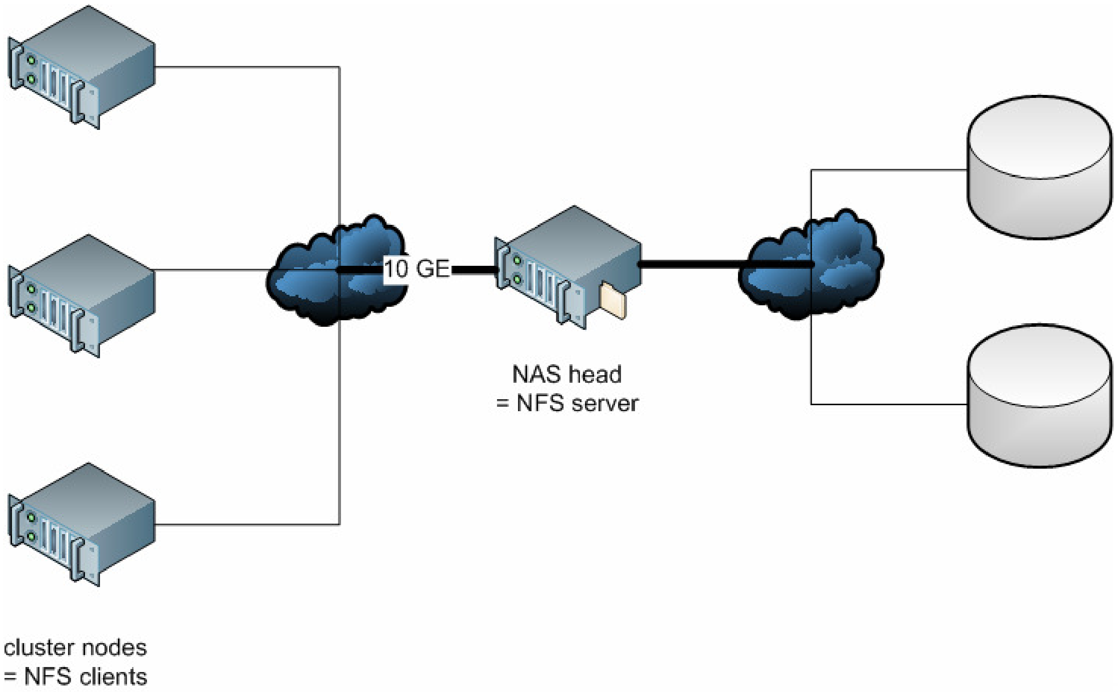
- actually yesterday's solution
- does not scale: NFS head is bottleneck



transtec

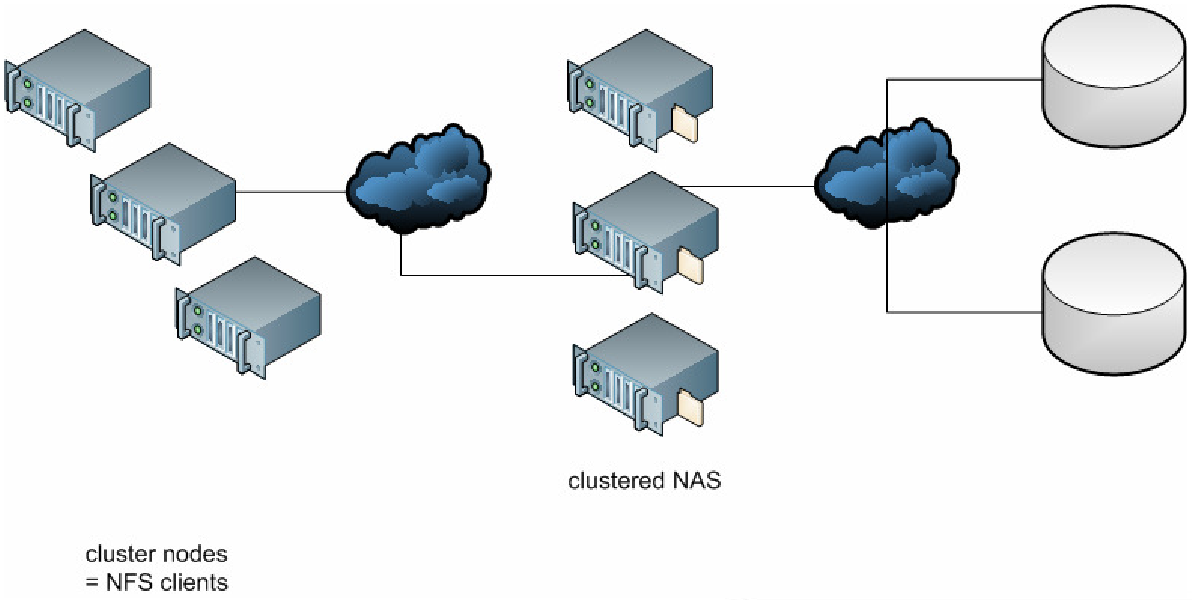
Solution with Short-Term Expiry Date: High-Speed NFS

- does not scale either
- NFS head will be bottleneck again by tomorrow



Problematic Enhancement: Clustered NFS

- either head-to-head synchronization limits scalability
- or manual partitioning of global namespace is cumbersome
- NFS is not suitable for dynamical load balancing (inherent state)



transtec

Distributed File Systems

- major features:
  - **global namespace** eases filesystem management and job flow
  - **scalable** capacities and bandwidths
  - **load balancing**
- **cluster** vs. **parallel filesystem:**
  - no shared storage → many-to-many access to data
- **proprietary solutions** already there:
  - IBM‘s GPFS
  - SGI‘s CXFS
  - Panasas‘ ActiveScale Filesystem (PanFS)
  - EMC‘s Celerra MPFS/MPFSi (pka High Road)
  - Lustre, PVFS2, ...

NFS as a Standard

- need for **OS independent, interoperable, standardized** solution

→ **NFS is the ONLY standard!**

- **standards are good, because...**
  - they **protect** end user **investment** in technology
  - they **ensure** a base level of **interoperability**
  - while at the same time **provide choice** among products
  - commonality leads to **less training, simpler deployment, higher acceptance...**

A Brief History of NFS (1)

- NFS originally designed by SUN in the 80's
- **NFS 3** now widely deployed
  - stateless by design, stateful in reality
  - a bunch of auxiliary protocols: NLM, NSM, MOUNT
  - 32 bit UIDs/GIDs
  - RPC procedure ACCESS for client-side access check
  - READDIRPLUS for efficient collection of file metadata within a directory
  - rsize/wsize of max 32k
- proprietarily extended on the quiet: WebNFS, ACLs, Secure RPC,...

A Brief History of NFS (2)

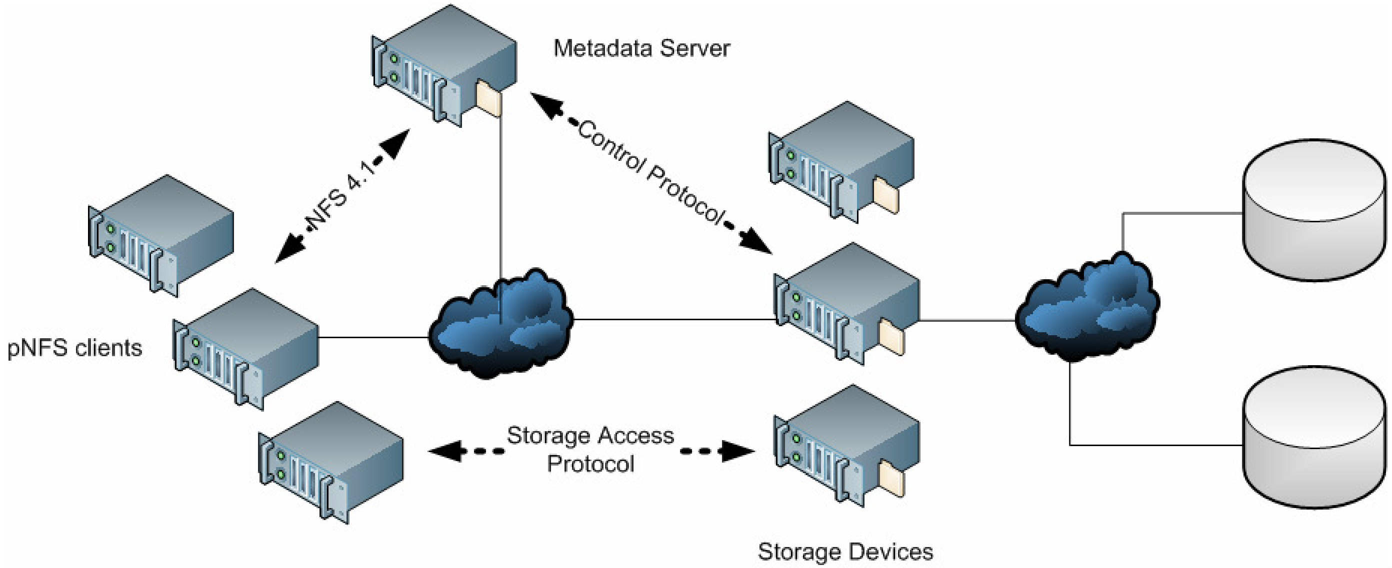
- **NFS 4:** under development from 1998-2005
  - primarily driven by Sun, Netapp, Hummingbird
  - some University involvement (CITI UMich, CMU)
  - now broadly available: Linux, Solaris, Windows, AIX,...
- lots of new stuff
  - **strong security flavors**: GSS_API (Kerberos, LIPKEY,...)
  - **protocol consolidation** (no NLM, NSM, MOUNT,...)
  - only one port: 2049 (**firewall friendly**)
  - **delegation** (cf. to CIFS oplocks)
  - **ACLs** (Windows-like)
  - **string-based identities**
  - **stateful** by design (lease-based state)
  - **COMPOUND** procedure for better performance

NFS 4.1 and Parallel NFS (pNFS)

- **NFS 4.1:** idea to use SAN FS architecture for NFS originally from Gary Grider (LANL) and Lee Ward (Sandia)
- development driven by Panasas, Netapp, Sun, EMC, IBM, UMich/CITI
- folded into NFSv4 minor version NFSv4.1 in 2006
- future **internet standard**
  (current draft 21: http://www.ietf.org/internet-drafts/draft-ietf-nfsv4-minorversion1-21.txt)
- major changes to NFS 4:
  - sessions
  - directory delegations
  - pNFS (optional feature)
- standardization expected some time in 2009

Parallel NFS (pNFS): Generic Architecture

- **separation** of **metadata** path and **data** path (**out-of-band** global namespace)
- built for **interoperability** and **backwards-compatibility**
- **flexible** design allows for different storage implementations (**layouts**)



What pNFS Does NOT Give You

- **improved cache consistency**
  - NFS has **open-to-close consistency**
- **perfect POSIX semantics** in a distributed file system
- **clustered metadata**
  - though a mechanism for this is not precluded

Parallel NFS (pNFS): New RPC Operations

- **`GETDEVICELIST (layouttype)`**
  - returns all device IDs for a specific file system
- **`GETDEVICEINFO (device_ID, layouttype)`**
  - returns the mapping of device ID to storage device address
- **`LAYOUTGET (layouttype, iomode, byterange)`**
  - returns file layout
- **`LAYOUTCOMMIT (filehandle, byterange, updated attributes, layout-specific info)`**
  - updated layout visible to other clients
  - timestamps, EOF attributes updated
- **`LAYOUTRETURN (filehandle, range)`**
  - releases state for client

Parallel NFS (pNFS): New RPC Callbacks

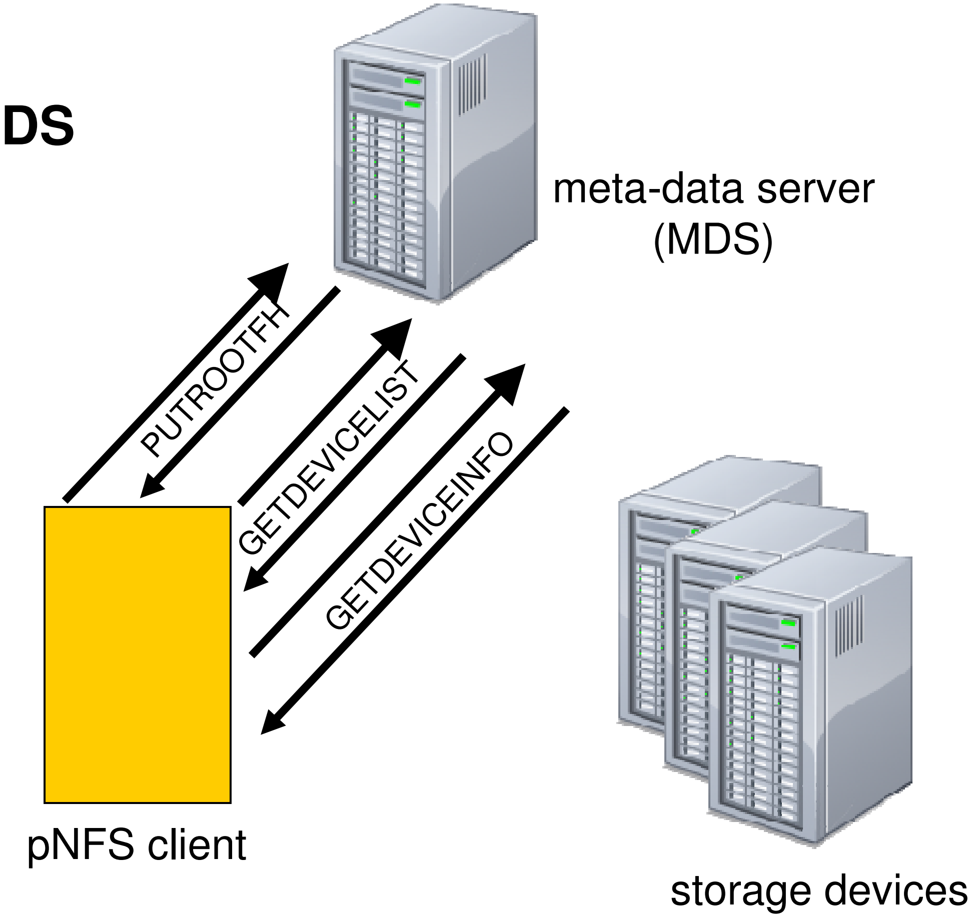
- **CB_LAYOUTRECALL**
  - tells a client to stop using a layout
- **CB_RECALL_ANY**
  - tells a client that it needs to return some number of recallable objects, including layouts
- **CB_RECALLABLE_OBJ_AVAIL**
  - delegation available for a layout that was not previously available
- **CB_NOTIFY_DEVICEID**
  - notifies the client of changes to device IDs

Parallel NFS (pNFS): How It Works (1)

- **clients mounts a filesystem via MDS**

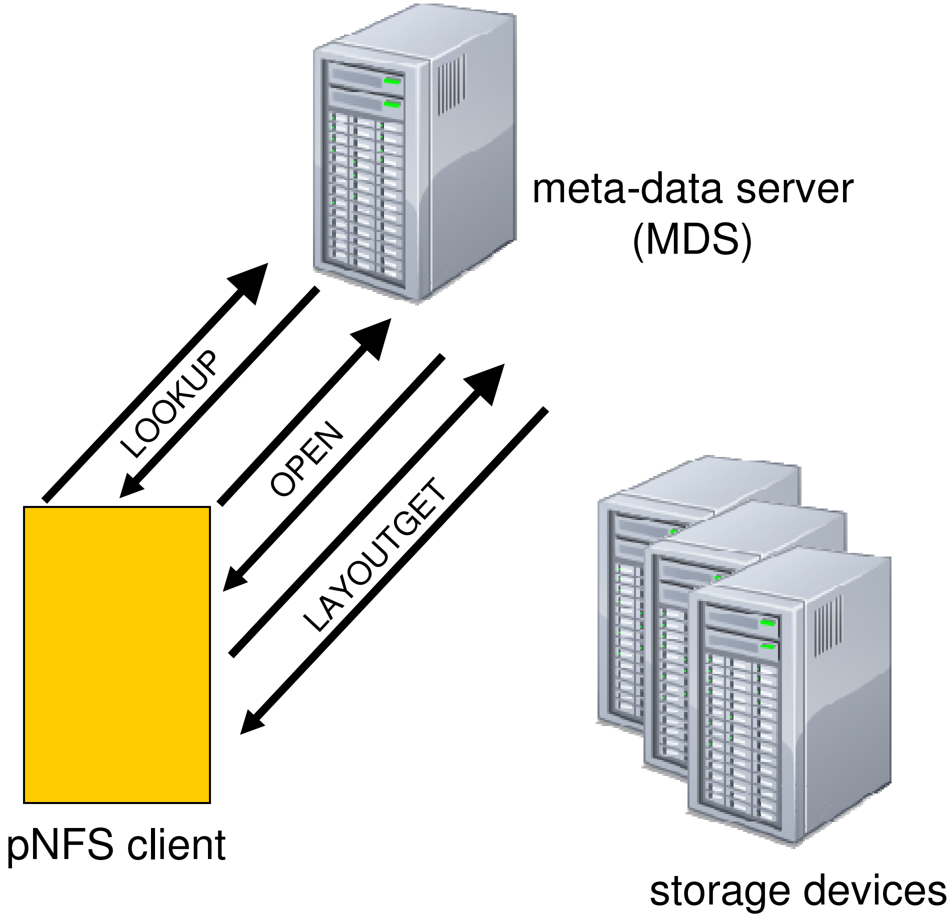
  `mount mds:/ /mnt`

  - client gets root filehandle from MDS
  - client gets list of device IDs for this filesystem (according to supported layouts)
  - client gets mapping of device IDs to storage device addresses



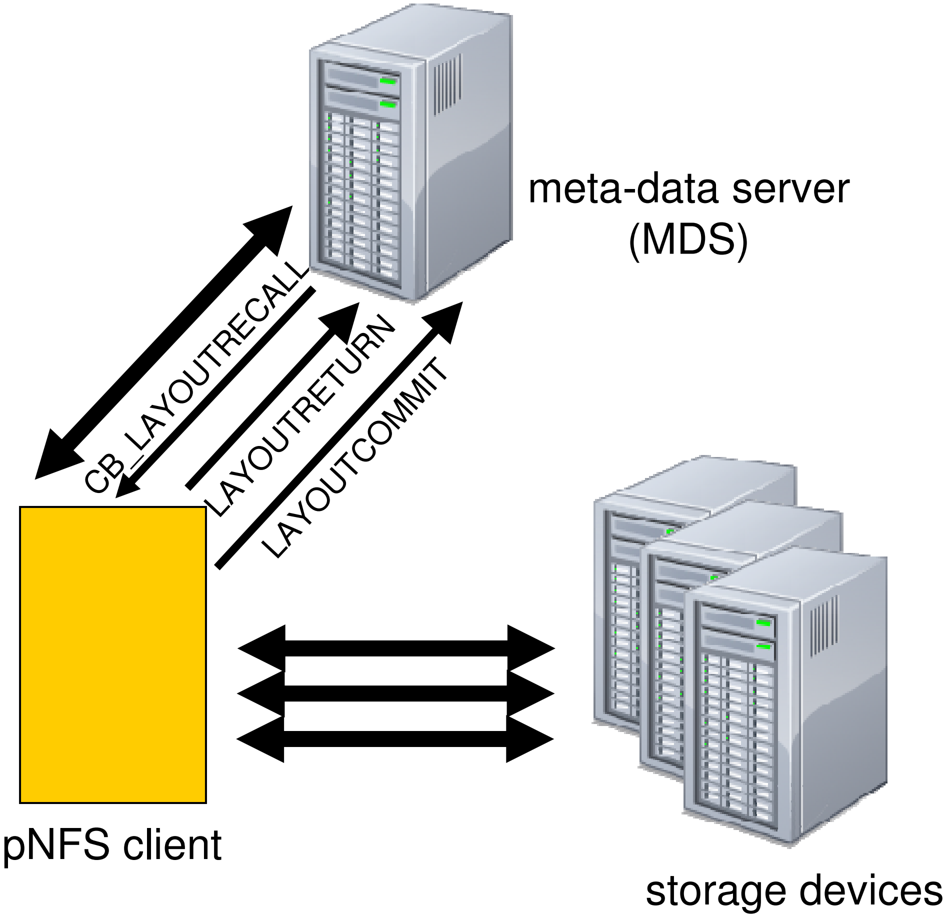
Parallel NFS (pNFS): How It Works (2)

- **clients looks up and opens a file**
  `fd = open(„/mnt/file“,...)`
  - client: looks up a file
  - server: returns file handle and state IDs
  - client: opens a file
  - client: asks MDS about layout for a file
  - server: hands over layout for file, containing device IDs and striping information



Parallel NFS (pNFS): How It Works (3)

- **client reads/writes to a file**
  `read/write (fd,...)`
  - client uses layout to perform I/O directly to storage devices (**READ/WRITE**)
    - at any time MDS can recall the layout
    - at any time client can return the layout
  - client commits changes and returns layout
- pNFS is optional, client can always use NFS 4 I/O via MDS

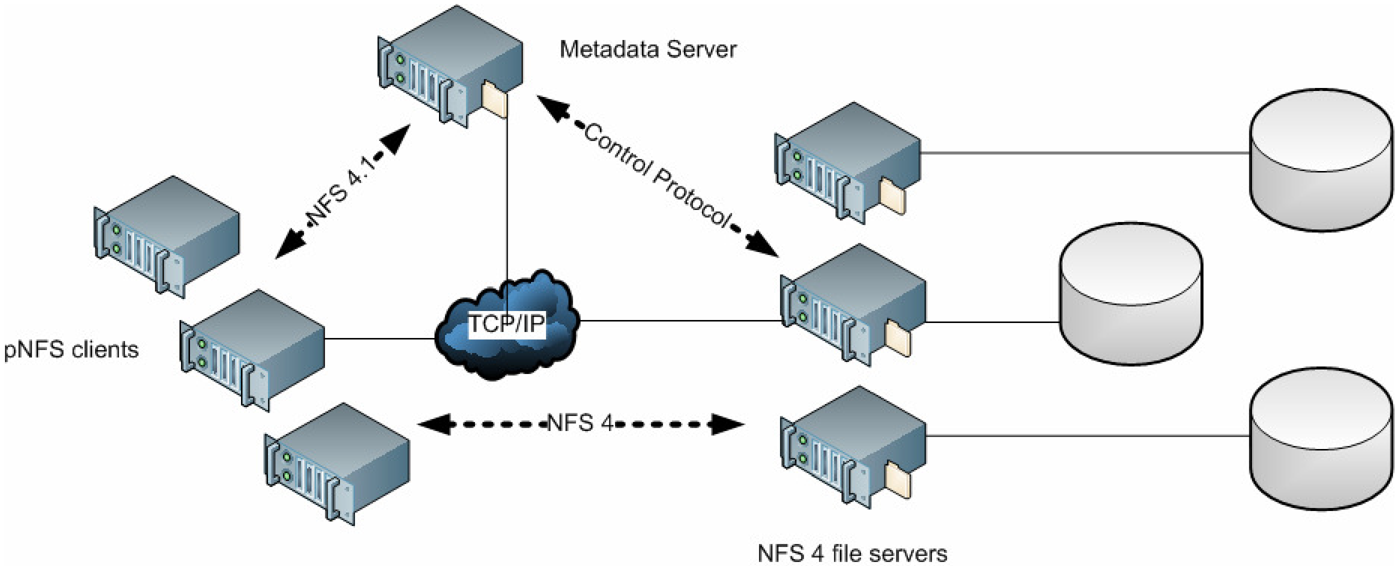


Parallel NFS (pNFS): Different Layout Formats

- a **layout** describes the location of file data, containing a list of device IDs and striping information
- possession of a layout grants access to storage devices, resp. files
- **file-based layout** (part of NFS 4.1/pNFS standard)
- **block-based layout:** http://www.ietf.org/internet-drafts/draft-ietf-nfsv4-pnfs-block-08.txt
- **object-based layout:** http://www.ietf.org/internet-drafts/draft-ietf-nfsv4-pnfs-obj-07.txt
- PVFS2 layout
- GPFS layout
- ...

pNFS: File Layout

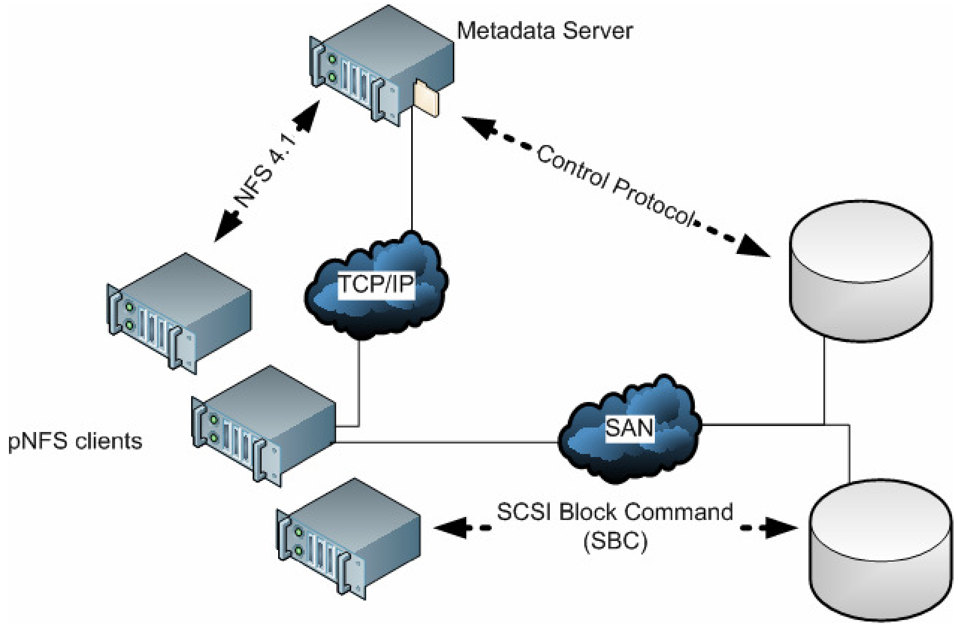
- only storage access protocol directly specified in NFS 4.1 standard
- significantly co-designed by NetApp, Sun, IBM and others
- file layout simple, may be **heavily cached** by clients
- access control possible via RPCSEC_GSS security flavor



pNFS: Block Layout

- highly influenced by **EMC's** design of **Multi-Path File System MPFS(i)** (pka **High Road)**
- block layout uses **volume identifiers, block offsets and extents**
- secure authorization with **host granularity only**, file-level security cannot be enforced by storage devices

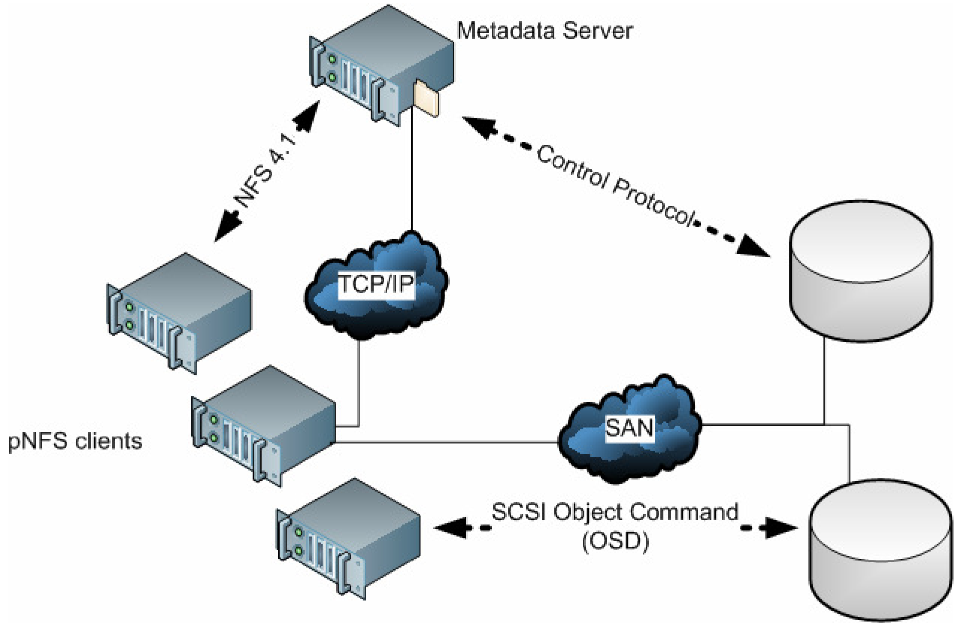
→ **clients must be trusted** (fundamental NFS problem ever since)



transtec

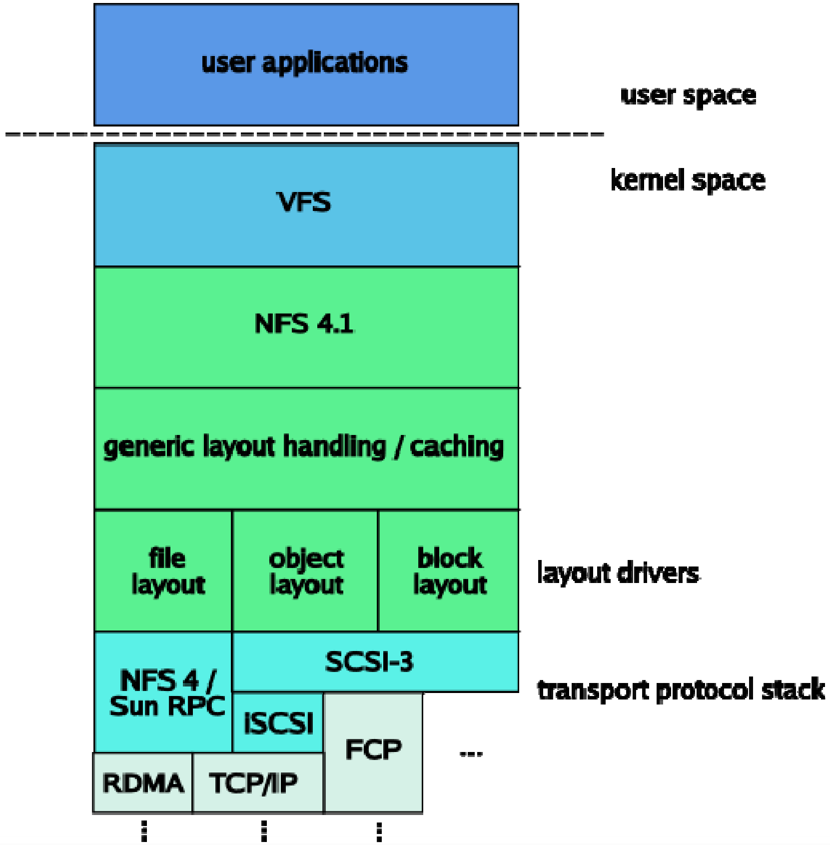
pNFS: Object Layout

- Panasas' contribution, based on **NASD** design (Network-Attached Secure Disk) developed at Carnegie Mellon University,
  later evolved into forthcoming **SCSI OSD** standard (**object-based storage device**)
- layout uses **SCSI object command** set
- **space management** built into devices
- designed for **secure access** and **high-performance data replication**
- cryptographically secured credentials ("**capabilities**") needed to access storage devices



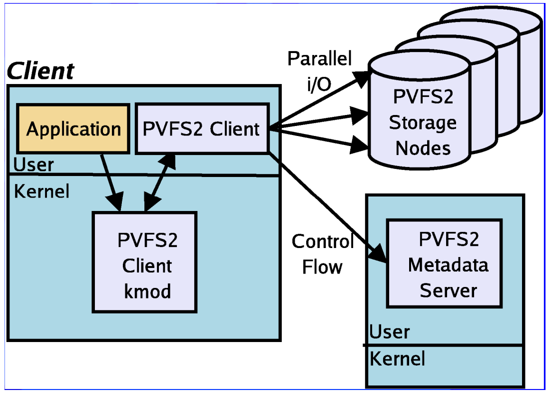
pNFS: Generic Implementation

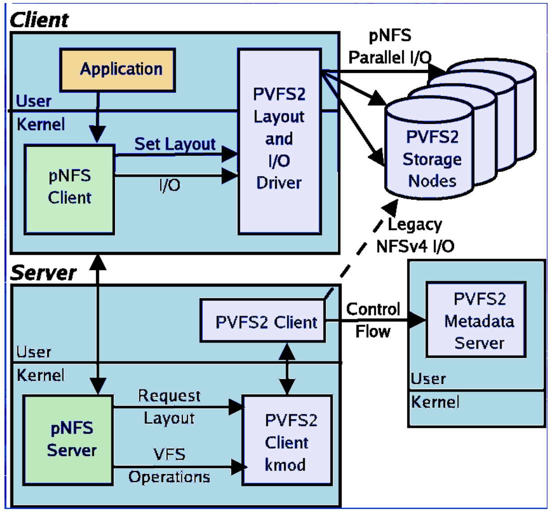
- modular and flexible design: manufacturers need only provide **layout drivers** to clients



pNFS: Linux Implementation

- prototype based on PVFS2: http://www.pvfs.org
  - developed at Argonne National Laboratory
  - algorithmic file layout, supports round robin striping (no LAYOUT<XXX>-Operations necessary)
  - no locking subsystem
  - no data caching
- pNFS server is PVFS2 client (pNFS↔PVFS2 proxy server)
- file layout driver will be completed soon: http://www.citi.umich.edu/projects/asci/pnfs/linux/
- block layout driver under development: http://www.citi.umich.edu/projects/nfsv4/pnfs/block/
- object layout: Panasas





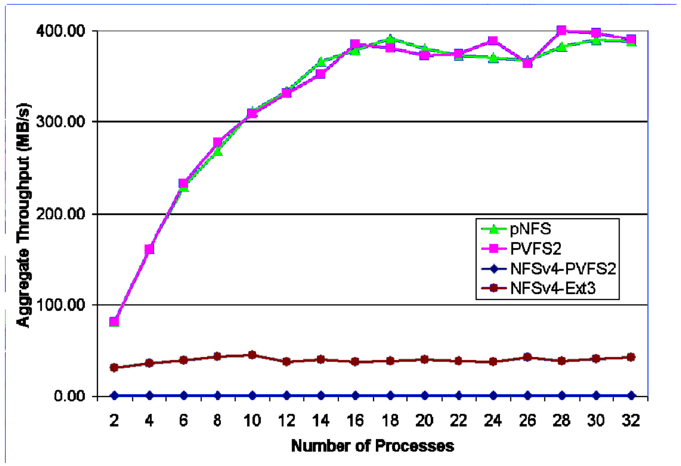
Source: www.citi.umich.edu

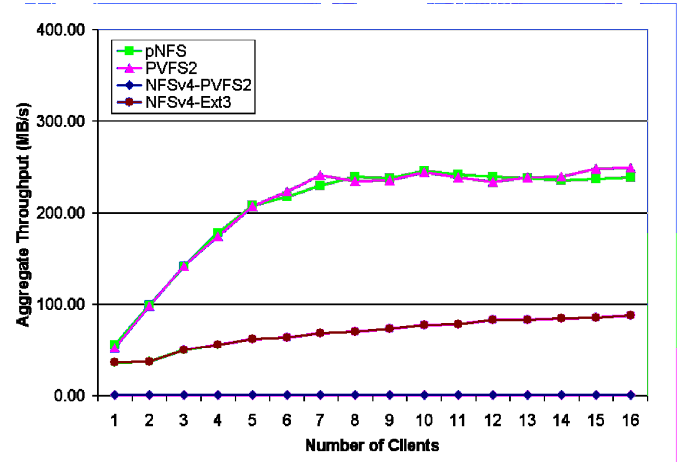
pNFS: The Current State

- **Linux:** file layout, based on PVFS2 / based on NFS 4
- **OpenSolaris:** file (NFS 4) / object (OSD-1) layout driver will be completed soon, patches available: http://opensolaris.org/os/project/nfsv41/ http://opensolaris.org/os/project/osd/
- **Netapp:** file layout, based on NFS 4
- **IBM:** file layout, based on GPFS
- **EMC:** block layout, based on MPFS(i)
- **Panasas:** object layout, based on ActiveScale PanFS
- **Carnegie Mellon University:** performance and correctness testing

Preliminary Benchmark Results: NFS vs. pNFS (1)

- source: http://www.citi.umich.edu/techreports/reports/citi-tr-05-1.pdf
- experimental setup:
  - 40 x 2 GHz Opteron nodes with 2 GB RAM each
  - 23 clients, 16 storage nodes, 1 MDS
  - RAID 0 for PVFS2
  - Gigabit-Ethernet
- write experiment:
  - (above) separate files
  - each client spawns 2 write processes
  - (below) single file

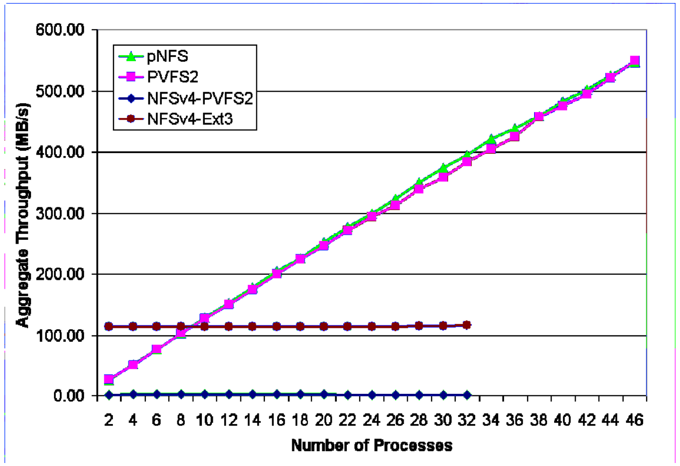


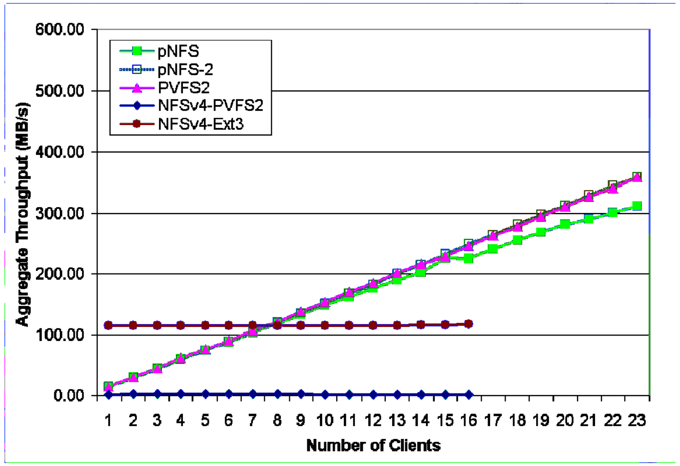


transtec

Preliminary Benchmark Results: NFS vs. pNFS (2)

- source: http://www.citi.umich.edu/techreports/reports/citi-tr-05-1.pdf
- experimental setup:
  - 40 x 2 GHz Opteron nodes with 2 GB RAM each
  - 23 clients, 16 storage nodes, 1 MDS
  - RAID 0 for PVFS2
  - Gigabit-Ethernet
- read experiment:
  - (above) separate files
  - each client spawns 2 read processes
  - (below) single file





Weblinks

- **NASD: Network Attached Secure Disks:** http://www.pdl.cmu.edu/NASD/
- **Panasas:** www.panasas.com
- **EMC Celerra Multi-Path File System:** http://www.emc.com/products/detail/software/celerra-multipath-file-system.htm
- **pNFS Information Portal:** http://www.pnfs.com
- **NFSv4 Status Pages:** http://tools.ietf.org/wg/nfsv4
- **Object-Based Storage Devices (now INCITS 400-2004):** http://www.t10.org/ftp/t10/drafts/osd/osd-r10.pdf
  Object-Based Storage Devices V2: http://www.t10.org/ftp/t10/drafts/osd2/osd2r03.pdf
- **Eisler's NFS Blog:** http://blogs.netapp.com/eislers_nfs_blog
- **NFSv4.1 Bakeathon at OpenSolaris.org:** http://opensolaris.org/os/project/nfsv41/nfsv41_bakeathon/

Thank you!

Backup

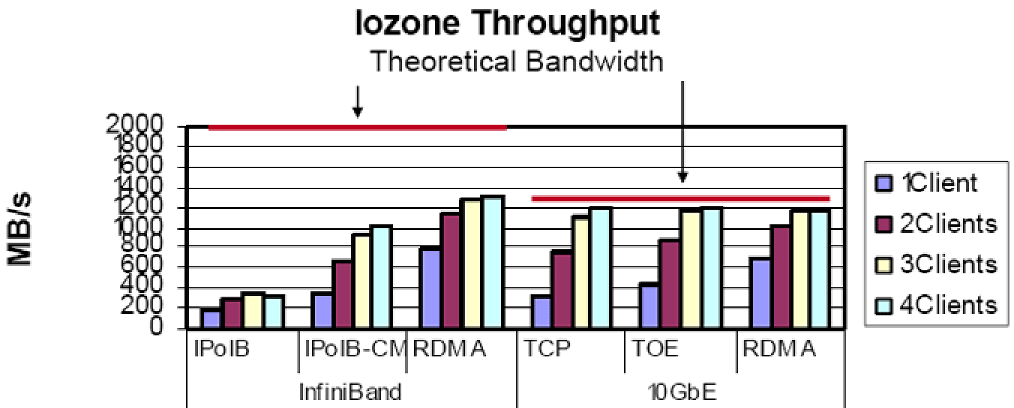
transtec

NFS Direct, InfiniBand, RDMA & All That...

- RDMA (Remote Direct Memory Access)
  - eliminates memory-to-memory copying (zero-copy)
  - OS bypass, low latency
  - http://www.rdmaconsortium.org
  - integrated into InfiniBand architecture
  - integrated into 10GE-RNICs with iWARP (iWARP = RDMA + TOE)
- iSER (iSCSI Extensions for RDMA)
  - additional transport layer for iSCSI communication (besides TCP)
- Linux RPC transport switch patches: http://oss.oracle.com/~cel/linux-2.6/
- Linux NFS/RDMA project: http://www.citi.umich.edu/projects/rdma/
- OpenSolaris NFS/RDMA: http://opensolaris.org/os/project/nfsrdma/

Preliminary Benchmark Results: RDMA vs. TCP/IP

- source: http://www.chelsio.com/nfs_over_rdma.html
- HW setup:
  - 1 NFS server, up to 4 clients
  - TCP/IPoIB-UD (MTU 2048), TCP/IPoIB-CM (MTU 65520), and IB RDMA transport at DDR
  - Host TCP/IP, TOE, and RNIC (iWARP) transport at 10GbE rate (MTU 9000)
- Results:
  - NFS over IB/RDMA slightly faster than 10 GbE
  - RDMA transport faster than TCP/IP
  - NFS over TCP
  - IPoIB-CM significantly better than IPoIB-UD

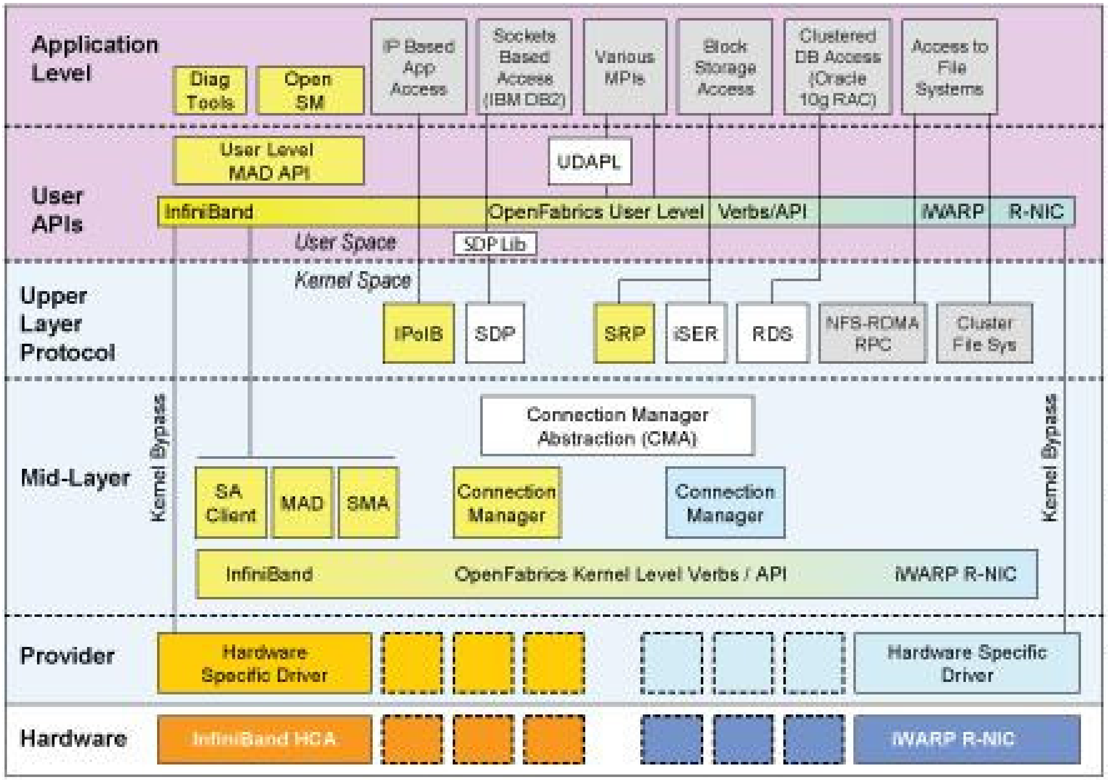


	IPoIB	IPoIB-CM	RDMA	TCP	TOE	RDMA
	InfiniBand			10GbE		
1Client	184.07	351.34	770.28	328.81	435.98	697.13
2Clients	294.18	655.37	1130.10	747.13	867.23	1025.59
3Clients	341.18	926.02	1274.64	1093.52	1146.62	1148.01
4Clients	329.11	1010.50	1307.39	1188.96	1191.07	1172.13

Transport

transtec

Today's Communication Protocol Stack

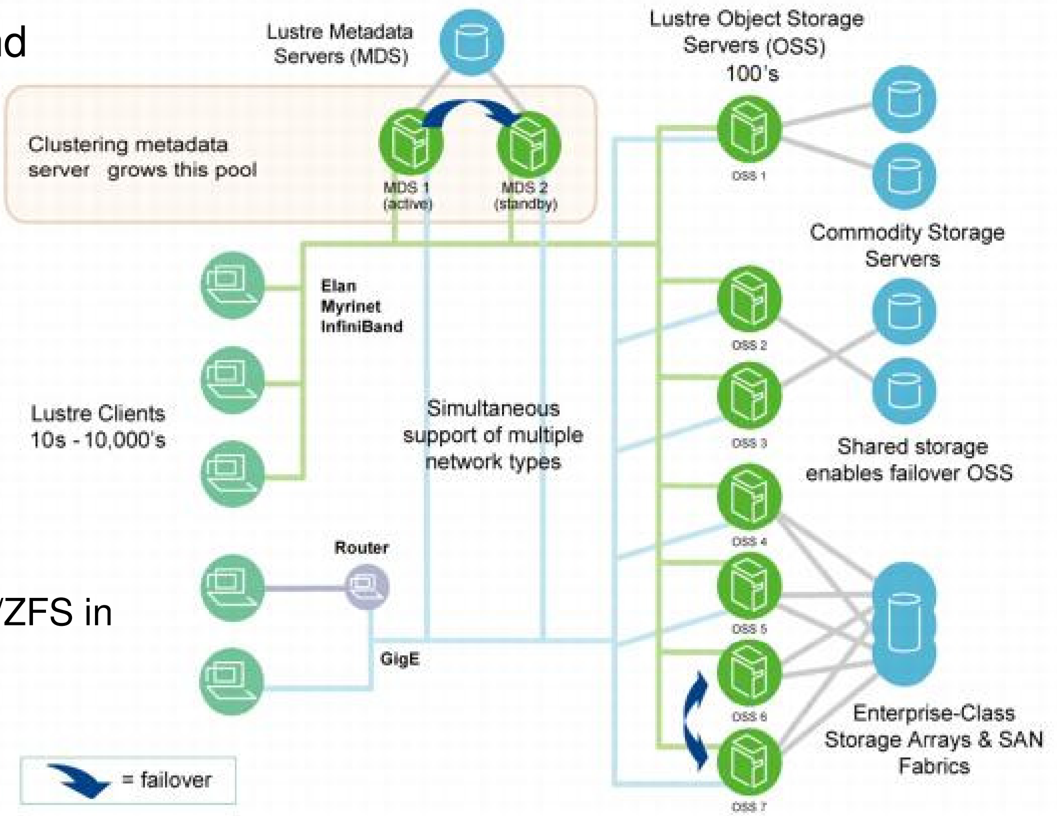


SA	Subnet Administrator
MAD	Management Datagram
SMA	Subnet Manager Agent
PMA	Performance Manager Agent
IPoIB	IP over InfiniBand
SDP	Sockets Direct Protocol
SRP	SCSI RDMA Protocol (Initiator)
iSER	iSCSI RDMA Protocol (Initiator)
RDS	Reliable Datagram Service
UDAPL	User Direct Access Programming Lib
HCA	Host Channel Adapter
R-NIC	RDMA NIC

Source: www.hpcwire.com

Other Projects: Lustre

- "Lustre" is a portmanteau of "Linux" and "cluster"
- originally developed by Cluster File Systems, Inc., acquired by Sun Microsystems, Inc. in 2007
- available under the GNU GPL
- integration with Linux/Ext3
- support for several high-speed interconnects
- future:
  - integration with Linux/Ext4 and Solaris/ZFS in userspace!
  - clustered metadata (in Lustre 2.0)
  - integration with pNFS design?



Source: www.lustre.org

Panasas / pNFS / OSD Roadmap