SDC
STORAGE DEVELOPER CONFERENCE
SNIA SANTA CLARA, 2008

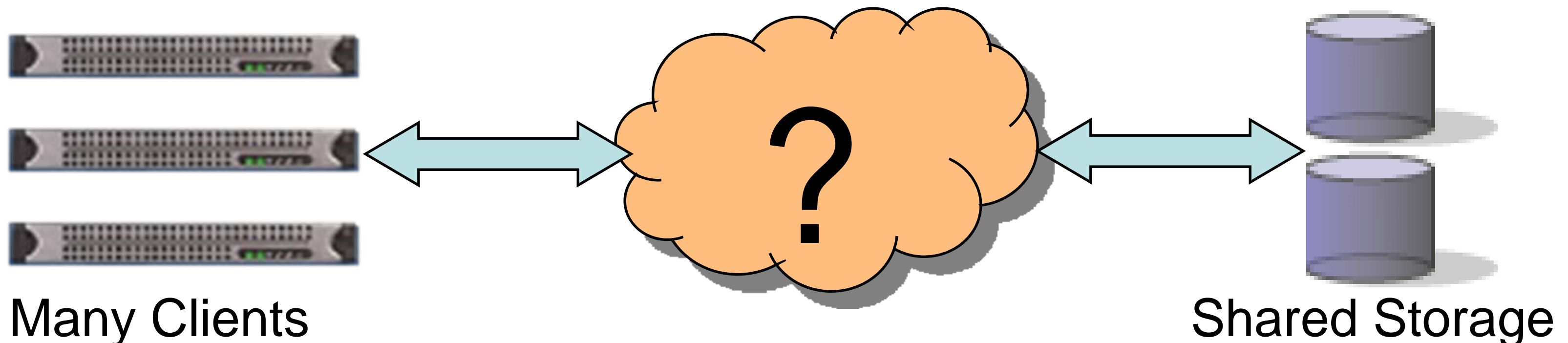
Clustered and Parallel Storage System Technologies

Brent Welch
Panasas, Inc.

© 2008 Panasas. All Rights Reserved.

Cluster Storage Problem Statement

- Compute clusters are grower larger in size (8, 128, 1024, 4096 nodes…)
  - Scientific codes, seismic data, digital animation, biotech, EDA …
- Each host in the cluster needs uniform access to any stored data
- Demand for storage capacity and bandwidth is growing (many GBs/sec)
- Apply clustering techniques to the storage system itself
- Maintain simplicity of storage management even at large scale



© 2008 Panasas. All Rights Reserved.

About Me

- Brent Welch
  - Director, Software Architecture at Panasas
  - IETF nfsv4 WG pNFS
  - Sun Labs (tclhttpd)
  - Xerox PARC (exmh)
  - UC Berkeley (Sprite)

© 2008 Panasas. All Rights Reserved.

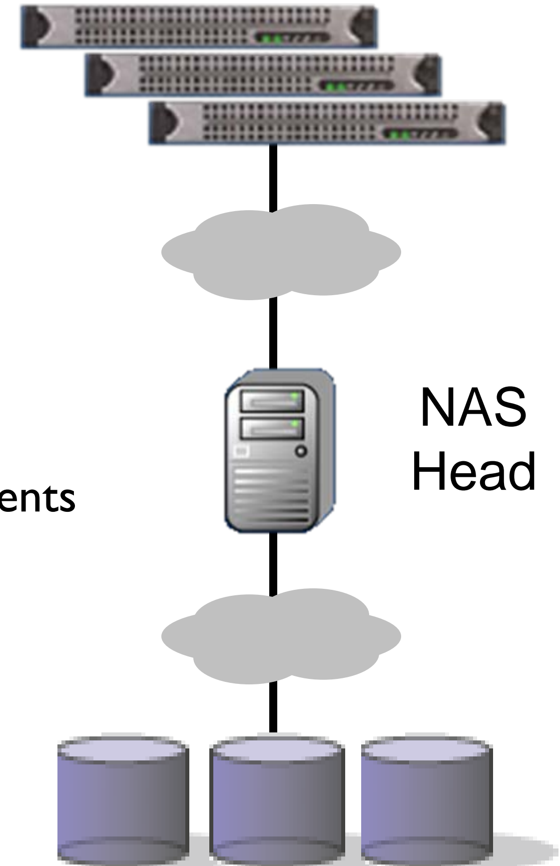
Parallel Filesystem Design Issues

- Have to solve same problems as local filesystem, at scale
  - Block allocation
  - Metadata management
  - Data reliability and error correction
- Additional requirements
  - Cache coherency
  - High availability
  - Scalable capacity & performance

© 2008 Panasas. All Rights Reserved.

Network Attached Storage (NAS)

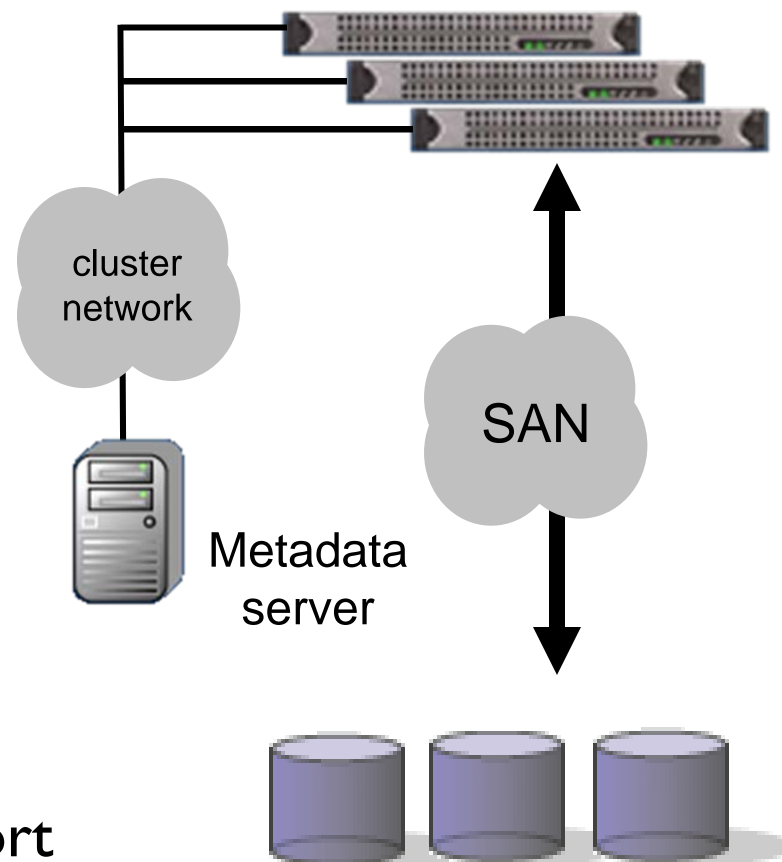
- File server exports local filesystem using a file-oriented protocol
  - NFS & CIFS are widely deployed
  - HTTP/WebDAV? FTP?
- Scalability limited by server hardware
  - Often the same CPU, RAM, I/O and memory buses as clients
  - Handles moderate number of clients
  - Handles moderate amount of storage
- A nice model until it runs out of steam
  - "Islands of storage"
  - Bandwidth to a file limited by server bottleneck



© 2008 Panasas. All Rights Reserved.

SAN Shared File Systems

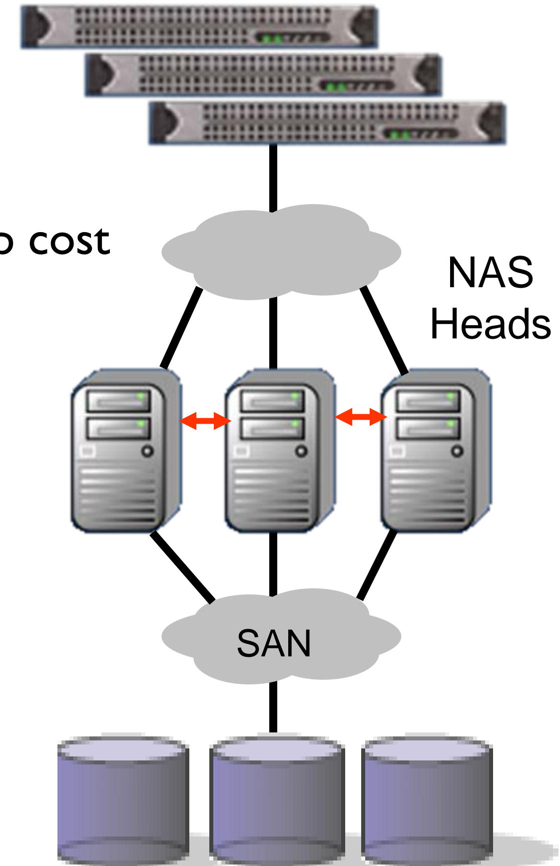
- SAN provides common management and provisioning for host storage
  - Block devices accessible via iSCSI or FC
  - Wire-speed performance potential
- Originally for local host FS
- Extended to create shared file system
  - Asymmetric (pictured): separate metadata server manages blocks and inodes
  - Symmetric: all nodes share metadata & block management
  - Reads & writes go direct to storage
- NAS access can be provided by "file head" client node(s) that re-export the SAN file system via NAS protocol
- IBM GPFS, Sun QFS, SGI CXFS



© 2008 Panasas. All Rights Reserved.

Clustered NAS

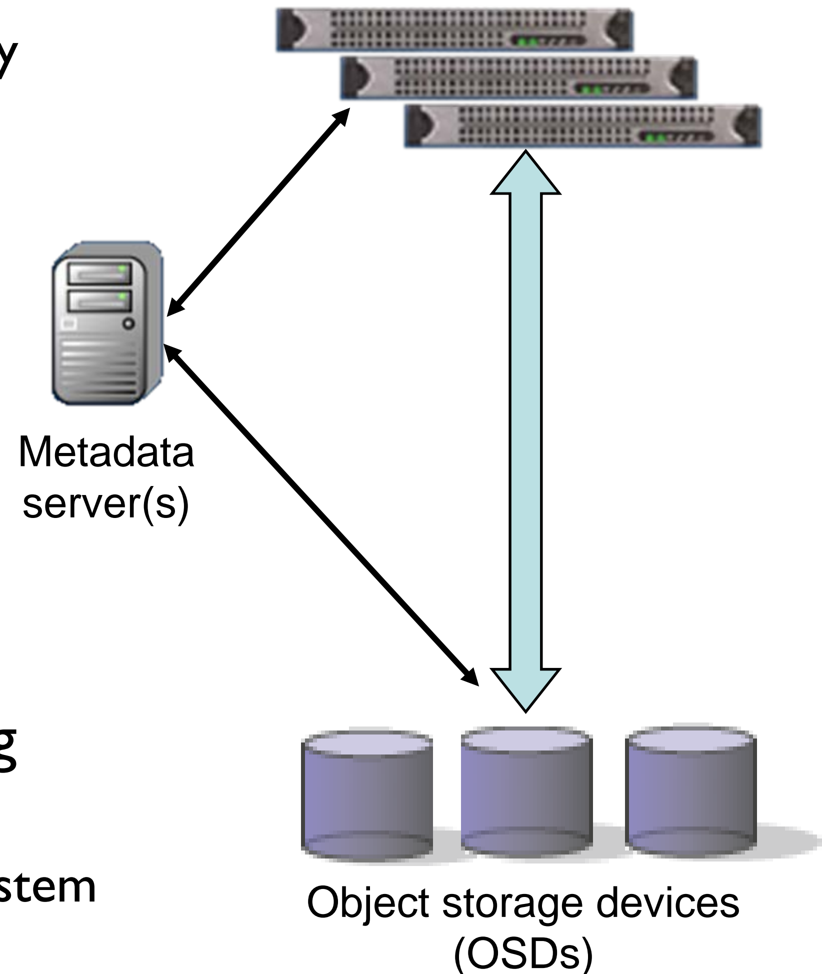
- More scalable than single-headed NAS
  - Multiple NAS heads share back-end storage
  - "In-band" NAS head still limits performance and drives up cost
- Two primary architectures
  - Export NAS from clustered SAN file system (pictured)
  - Private storage, forward requests to owner
- NFS shortcomings for HPC
  - No good mechanism for dynamic load balancing
  - Poor coherency (unless you disable client caching)
  - No parallel access to data (until pNFS)
- Isilon OneFS, NetApp GX, BlueArc, AFS



© 2008 Panasas. All Rights Reserved.

Object-based Storage Clusters

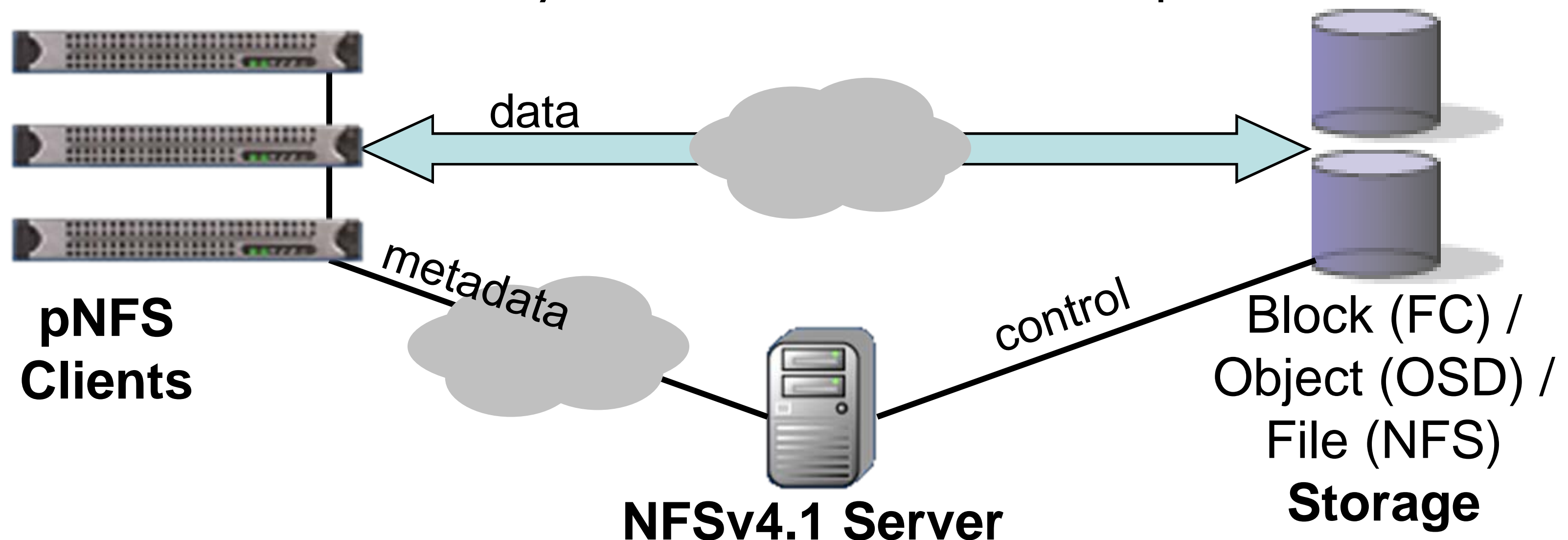
- Object Storage Devices
  - High-level interface that includes security
  - Block management inside the device
  - OSD standard (SCSI T10)
- File system layered over objects
  - Metadata server manages namespace and external security
  - OSD manages block allocation and internal security
  - Out-of-band data transfer directly between OSDs and clients
- High performance through clustering
  - Scalable to thousands of clients
  - >> GB/sec demonstrated to single filesystem



© 2008 Panasas. All Rights Reserved.

pNFS: Standard Storage Clusters

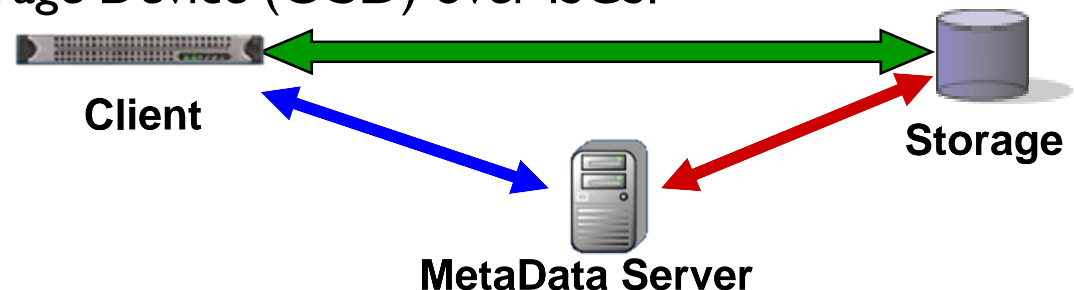
- pNFS is an extension to the Network File System v4 protocol standard
- Allows for parallel and direct access
  - From Parallel Network File System clients
  - To Storage Devices over multiple storage protocols
  - Moves the Network File System server out of the data path



© 2008 Panasas. All Rights Reserved.

The pNFS Standard

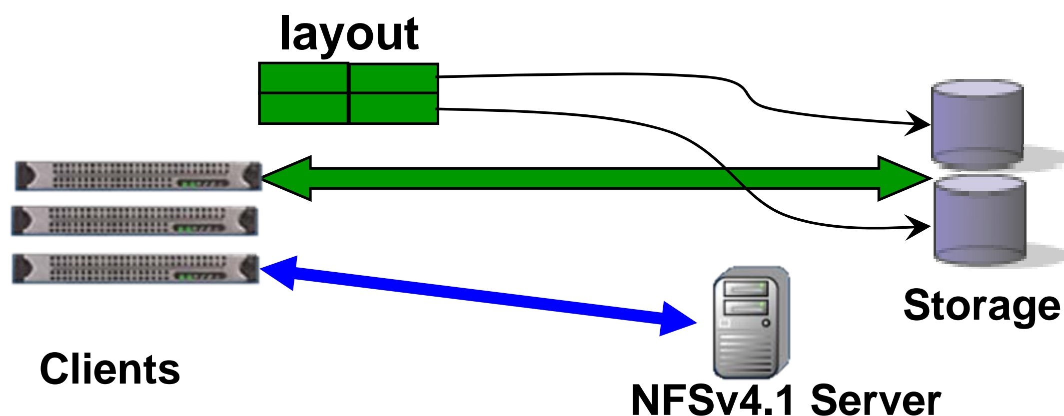
- The pNFS standard defines the NFSv4.1 protocol extensions between the server and client
- The I/O protocol between the client and storage is specified elsewhere, for example:
  - SCSI **Block** Commands (**SBC**) over Fibre Channel (**FC**)
  - SCSI **Object**-based Storage Device (**OSD**) over iSCSI
  - Network **File** System (**NFS**)
- The control protocol between the server and storage devices is also specified elsewhere, for example:
  - SCSI **Object**-based Storage Device (**OSD**) over iSCSI



© 2008 Panasas. All Rights Reserved.

pNFS Layouts

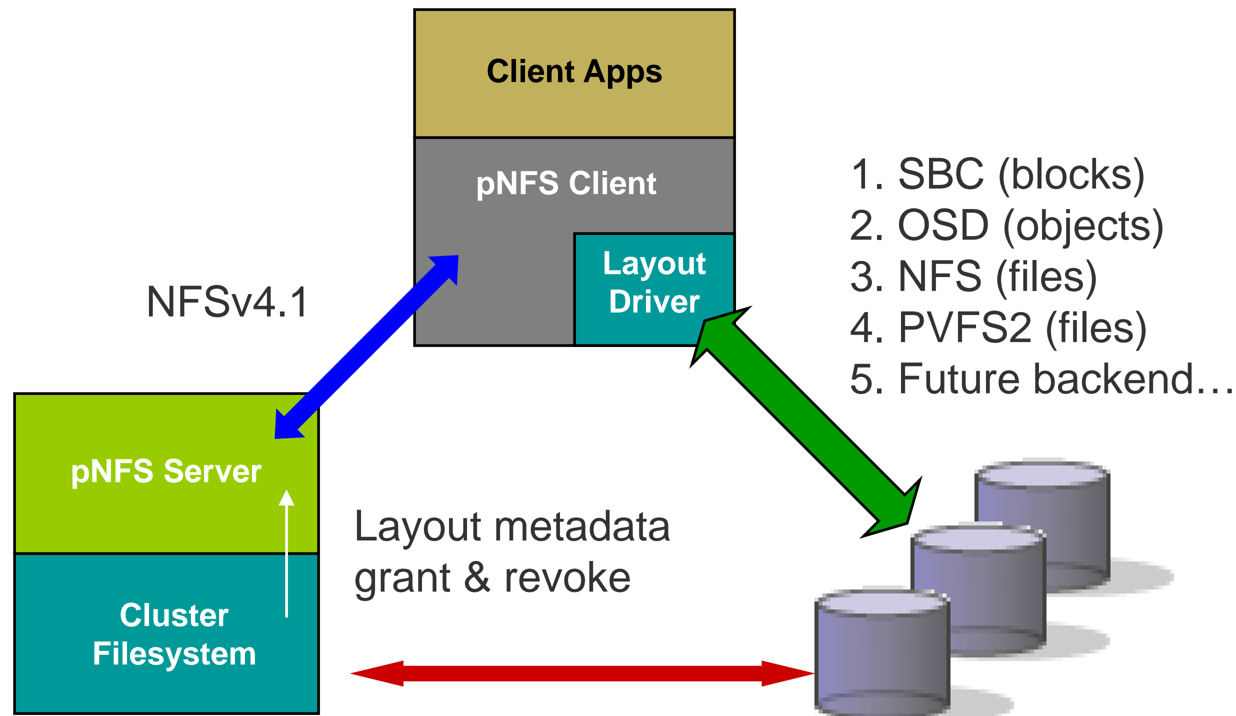
- Client gets a *layout* from the NFS Server
- The layout maps the file onto storage devices and addresses
- The client uses the layout to perform direct I/O to storage
- At any time the server can recall the layout
- Client commits changes and returns the layout when it's done
- pNFS is optional, the client can always use regular NFSv4 I/O



© 2008 Panasas. All Rights Reserved.

pNFS Client

- Common client for different storage back ends
- Wider availability across operating systems
- Fewer support issues for storage vendors



© 2008 Panasas. All Rights Reserved.

pNFS is not…

- Improved cache consistency
  - NFS has open-to-close consistency enforced by client polling of attributes
  - NFSv4.1 directory delegations can reduce polling overhead
- Perfect POSIX semantics in a distributed file system
  - NFS semantics are good enough (or, all we'll give you)
  - But note also the POSIX High End Computing Extensions Working Group
    - http://www.opengroup.org/platform/hecewg/
- Clustered metadata
  - Not a server-to-server protocol for scaling metadata
  - But, it doesn't preclude such a mechanism

© 2008 Panasas. All Rights Reserved.

pNFS Protocol Operations

- LAYOUTGET
  - (filehandle, type, byte range) -> type-specific layout
- LAYOUTRETURN
  - (filehandle, range) -> server can release state about the client
- LAYOUTCOMMIT
  - (filehandle, byte range, updated attributes, layout-specific info) -> server ensures that data is visible to other clients
  - Timestamps and end-of-file attributes are updated
- GETDEVICEINFO, GETDEVICELIST
  - Map deviceID in layout to type-specific addressing information

© 2008 Panasas. All Rights Reserved.

pNFS Protocol Callback Operations

- Stateful NFSv4 servers make callbacks to their clients
- CB_LAYOUTRECALL
  - Server tells the client to stop using a layout
- CB_RECALLABLE_OBJ_AVAIL
  - Delegation available for a file that was not previously available

© 2008 Panasas. All Rights Reserved.

Parallel File Systems in Practice

GPFS

PVFS

Lustre

© 2008 Panasas. All Rights Reserved.

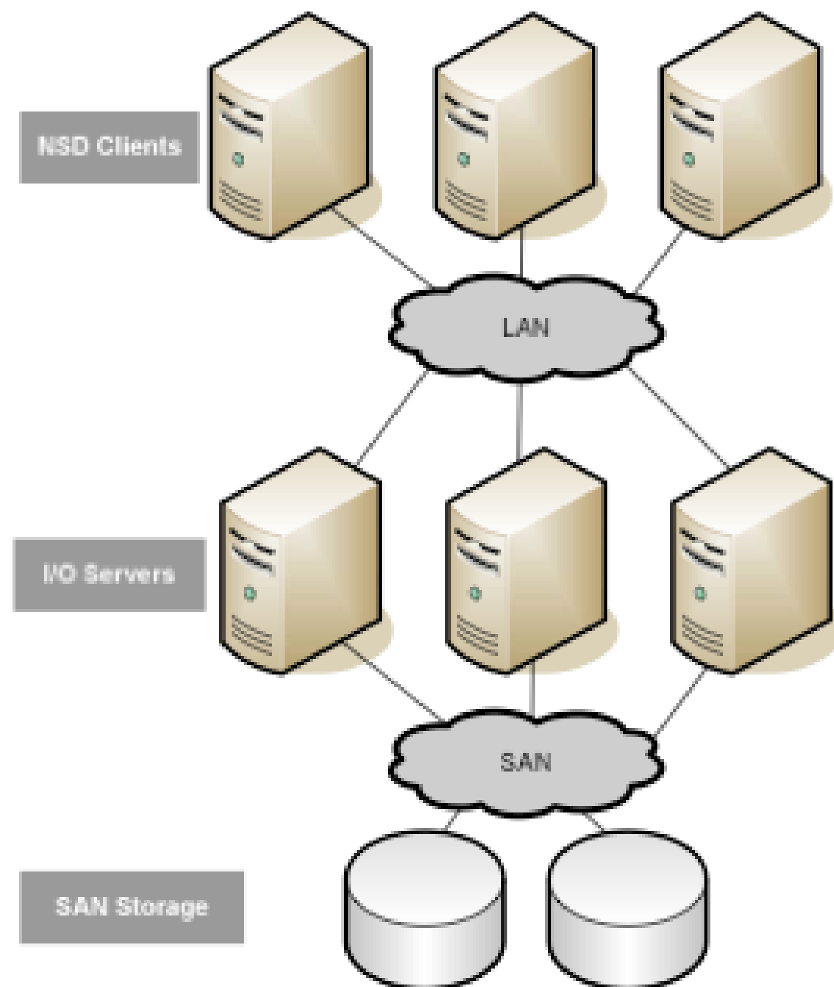
Production Parallel File Systems

- All 4 systems scale to support the very largest compute clusters (LLNL Purple, LANL RoadRunner, Sandia Red Storm, etc.)
- All delegate block management to “object-like” data servers
- Approaches to metadata vary
- Approaches to fault tolerance vary
- Emphasis on features, “turn-key” deployment, vary

© 2008 Panasas. All Rights Reserved.

IBM GPFS

- IBM General Parallel File System
- Legacy: IBM Tiger multimedia filesystem
- Commercial product
- Lots of configuration flexibility
  - AIX, SP3, Linux
  - Direct storage, Virtual Shared Disk, Network Shared Disk
  - Clustered NFS re-export
- Block interface to storage nodes
- Distributed locking



© 2008 Panasas. All Rights Reserved.

GPFS: Block Allocation

- I/O server exports exports local disk via block-oriented NSD protocol
- Block allocation map shared by all nodes
  - Block map split into N regions
  - Each region has 1/Nth of the blocks on each I/O server
- Writing node performs block allocation
  - Locks a region of the block map to find free blocks
  - Updates inode & indirect blocks
  - Making number of regions match number of client nodes reduces block map sharing collisions
- Stripe each block across multiple I/O servers (RAID-0)
- Large block size (1-4 MB) typically used
  - Increases transfer size per I/O server
  - Minimizes block allocation overhead
  - Not great for small files

© 2008 Panasas. All Rights Reserved.

GPFS: Metadata management

- Symmetric model with distributed locking
- Each node acquires locks and updates metadata structures itself
- Global token manager manages locking assignments
  - Client accessing a shared resource contacts token manager
  - Token manager gives token to client, or tells client current holder of token
  - Token owner manages locking, etc. for that resource
  - Client acquires read/write lock from token owner before accessing resource
  - Sharing patterns can cause “fluttering” of token ownership
- inode updates optimized for multiple writers
  - Shared write lock on inode
  - “Metanode token” for file controls which client writes inode to disk
  - Other clients send inode updates to metanode, which merges them

© 2008 Panasas. All Rights Reserved.

GPFS: Caching

- Client caching with strong coherency
- Client acquires read/write lock before accessing data
- Optimistic locking algorithm
  - First node accesses [0,1024), locks 0…EOF
  - Second node accesses [1024,2048)
    - First node reduces its lock to 0…1024
    - Second node locks 1024…EOF
  - Lock splitting assumes client will continue accessing in current pattern (forward or backward sequential)
- Client cache maintained separately from OS page/buffer cache

© 2008 Panasas. All Rights Reserved.

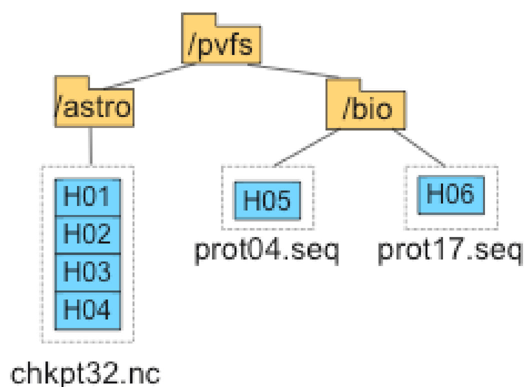
GPFS: Reliability

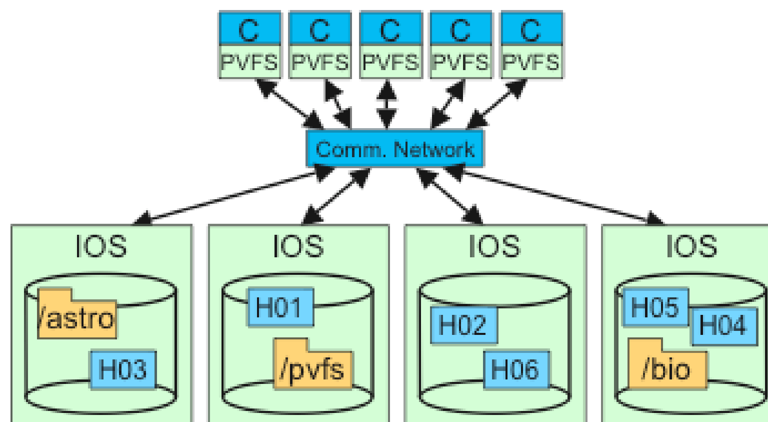
- RAID underneath I/O server to handle disk failures & sector errors
- Replication across I/O servers supported, but typically only used for metadata
- I/O server failure handled via dual-attached RAID or SAN
  - Backup I/O server takes over primary's disks if it fails
  - Can designate up to 8 potential owners for a disk (serial failover)
- Nodes journal metadata updates before modifying FS structures
  - Journal is per-node, so no sharing/locking issues
  - Journal kept in shared storage (i.e., on the I/O servers)
  - If node crashes, another node replays its journal to make FS consistent
- Quorum/consensus protocol to determine set of "online" nodes
  - Disk leases or SCSI-3 persistent reservations used for fencing

© 2008 Panasas. All Rights Reserved.

PVFS2

- Parallel Virtual Filesystem v2
- Open source
- Linux based
- Development led by Argonne National Lab
  - Supported by many other institutions & companies
- Asymmetric architecure (data servers & clients)
- Data servers use object-like API
- Focus on needs of HPC applications
  - Interface optimized for MPI-IO semantics, not POSIX





© 2008 Panasas. All Rights Reserved.

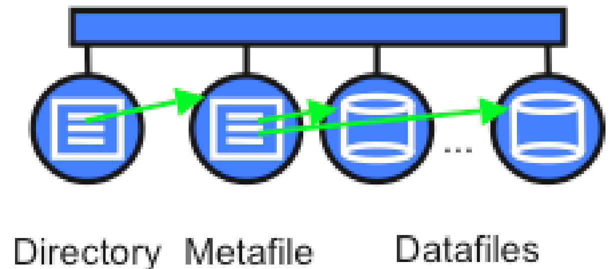
PVFS2: Block Allocation

- I/O server exports file/object oriented API
  - Storage object ("dataspace") on an I/O server addressed by numeric handle
  - Dataspace can be stream of bytes or key/value pairs
  - Create dataspace, delete dataspace, read/write
- Files & directories mapped onto dataspaces
  - File may be single dataspace, or chunked/striped over several
- Each I/O server manages block allocation for its local storage
- I/O server uses local filesystem (ext3, XFS, etc.) to store dataspaces
- Key/value dataspace stored using Berkeley DB table

© 2008 Panasas. All Rights Reserved.

PVFS2: Metadata management

- Directory dataspace contains list of names & metafile handles
- Metafile dataspace contains
  - Attributes (permissions, owner, xattrs)
  - Distribution function parameters
  - Datafile handles



- Datafile(s) store file data
  - Distribution function determines pattern
  - Default is 64 KB chunk size and round-robin placement
- Directory and metadata updates are atomic
  - Eliminates need for locking
  - May require "losing" node in race to do significant cleanup
- System configuration (I/O server list, etc.) stored in static file on all I/O servers

© 2008 Panasas. All Rights Reserved.

PVFS2: Caching

- Client only caches immutable metadata and read-only files
- All other I/O (reads, writes) go through to I/O node
- Strong coherency (writes are immediately visible to other nodes)
- Flows from PVFS2 limitations
  - No locking
  - No cache coherency protocol
- I/O server can cache data & metadata for local dataspaces
- All prefetching must happen on I/O server
- Reads & writes will never go faster than client's interconnect

© 2008 Panasas. All Rights Reserved.

PVFS2: Reliability

SDC STORAGE DEVELOPER CONFERENCE SNIA SANTA CLARA, 2008

- Similar to GPFS
  - RAID underneath I/O server to handle disk failures & sector errors
  - Dual attached RAID to primary/backup I/O server to handle I/O server failures
- Linux HA used for generic failover support
  - Remote control power strip (STONITH) for fencing
- Sequenced operations provide well-defined crash behavior
  - Example: Creating a new file
    - Create datafiles
    - Create metafile that points to datafiles
    - Link metafile into directory (atomic)
  - Crash can result in orphans, but no other inconsistencies

© 2008 Panasas. All Rights Reserved.

Panasas ActiveScale (PanFS)

- Commercial product based on CMU NASD research
- Complete “appliance” solution (HW + SW), blade server form factor
  - DirectorBlade = metadata server
  - StorageBlade = OSD
- Coarse grained metadata clustering
- Linux native client for parallel I/O
- NFS & CIFS re-export
- Integrated battery/UPS
- Global namespace

© 2008 Panasas. All Rights Reserved.

PanFS: Block Allocation

- OSD exports object-oriented API
  - Objects have a number (object ID), data, and attributes
  - CREATE OBJECT, REMOVE OBJECT, READ, WRITE, GET ATTRIBUTE, SET ATTRIBUTE, etc.
  - Commands address object ID and data range in object
  - Capabilities provide fine-grained revocable access control
- OSD manages private local storage
  - Two SATA drives, 500/750/1000 GB each, 1-2 TB total capacity
- Specialized filesystem (OSDFS) stores objects
  - Delayed floating block allocation
  - Efficient copy-on-write support
- Files and directories stored as "virtual objects"
  - Virtual object striped across multiple container objects on multiple OSDs

© 2008 Panasas. All Rights Reserved.

PanFS: Metadata management

- Directory is a list of names & object IDs in a RAID-1 virtual object
- Filesystem metadata stored as object attributes
  - Owner, ACL, timestamps, etc.
  - Layout map describing RAID type & OSDs that hold the file
- Metadata server (DirectorBlade)
  - Checks client permissions & provides map/capabilities
  - Performs namespace updates & directory modifications
  - Performs most metadata updates
- Client modifies some metadata directly (length, timestamps)
- Coarse-grained metadata clustering based on directory hierarchy

© 2008 Panasas. All Rights Reserved.

PanFS: Caching

- Clients cache reads & writes
- Strong coherency, based on callbacks
  - Client registers callback with metadata server
  - Callback type identifies sharing state (unshared, read-only, read-write)
  - Server notifies client when file or sharing state changes
- Sharing state determines caching allowed
  - Unshared: client can cache reads & writes
  - Read-only shared: client can cache reads
  - In read-write shared: client caching disabled
  - Specialized "concurrent write" mode for cooperating apps (e.g. MPI-IO)
- Client cache shared with OS page/buffer cache

© 2008 Panasas. All Rights Reserved.

PanFS: Reliability

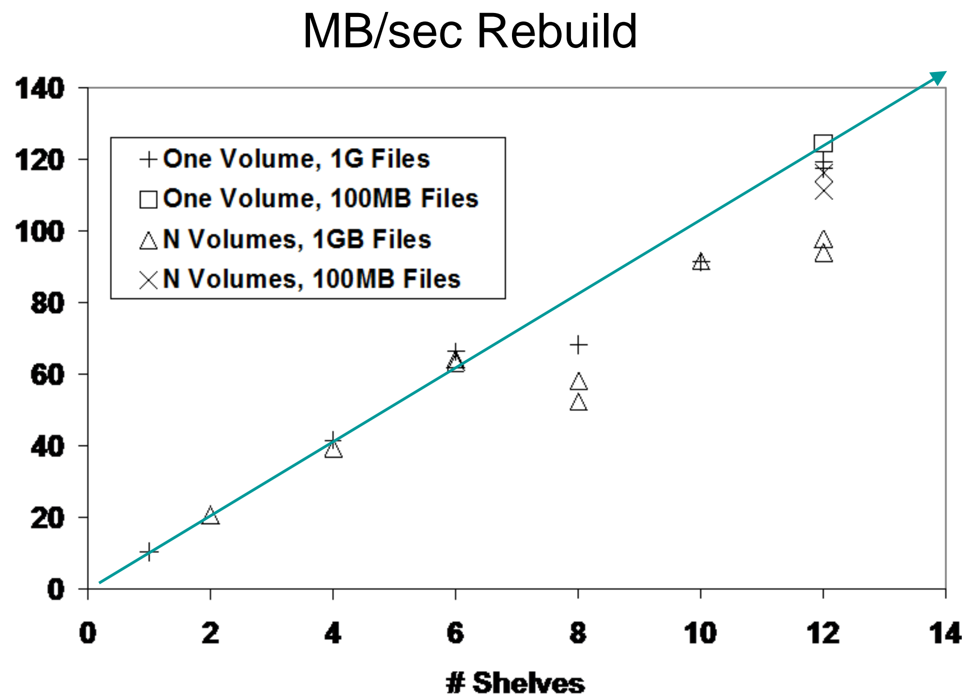
- RAID-1 & RAID-5 across OSDs to handle disk failures
  - Any failure in StorageBlade (disk, RAM, CPU) is handled via rebuild
  - Declustered parity allows scalable rebuild
- "Vertical parity" inside OSD to handle sector errors
- Integrated shelf battery makes all RAM in blades into NVRAM
  - Metadata server journals updates to in-memory log
    - Failover config replicates log to 2nd blade's memory
    - Log contents saved to DirectorBlade's local disk on panic or power failure
  - OSDFS commits updates (data+metadata) to in-memory log
    - Log contents committed to filesystem on panic or power failure
    - Disk writes well ordered to maintain consistency
- System configuration in replicated database on DirectorBlades

© 2008 Panasas. All Rights Reserved.

Panasas Scalable Rebuild

- Two main causes of RAID failures
  1) **2nd drive failure in same RAID set during reconstruction of 1st failed drive**
     - Risk of two failures depends on time-to-repair
  2) Media failure in same RAID set during reconstruction of 1st failed drive

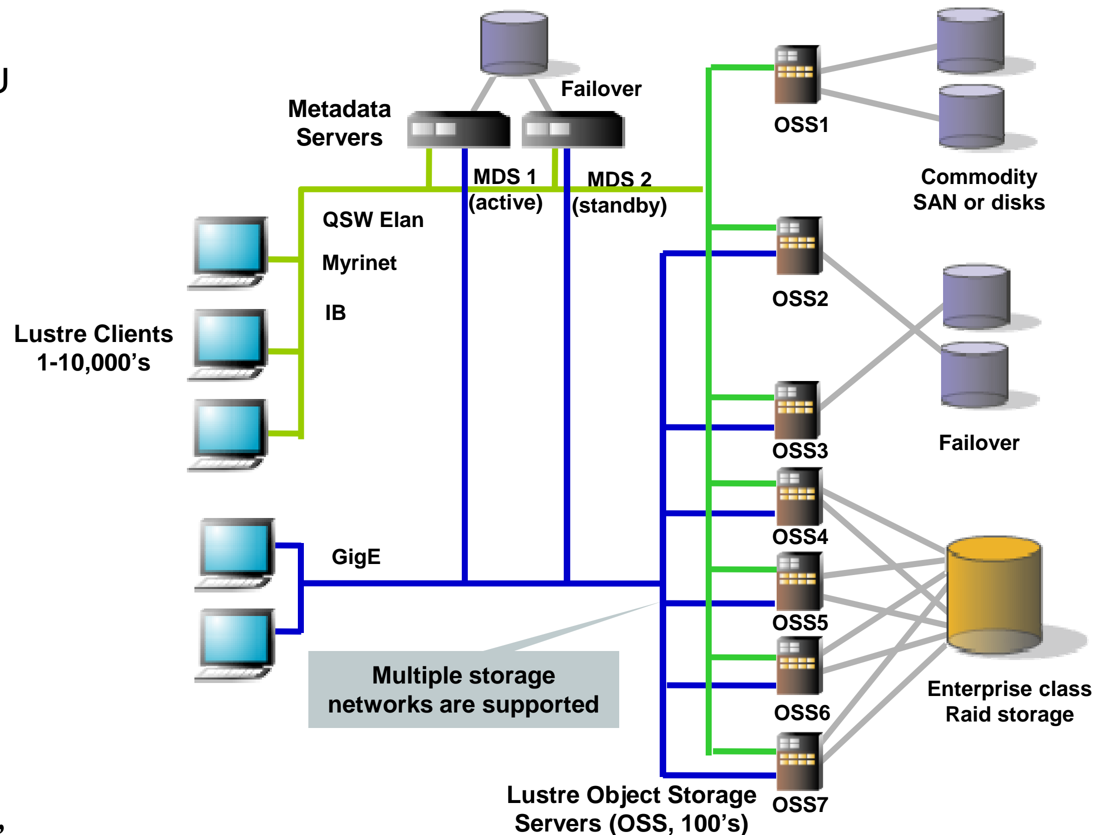
- Shorter repair time in larger storage pools
  - From 13 hours to 30 minutes
- Four techniques to reduce MTTR
  - Use multiple "RAID engines" (DirectorBlades) in parallel
  - Spread disk I/O over more disk arms (StorageBlades)
  - Reconstruct data blocks only – not unused space
  - Proactively remove failing blades (SMART trips, other heuristics)



© 2008 Panasas. All Rights Reserved.

Lustre (Object-based Storage)

- Open source object-based storage system
  - Based on NASD architecture from CMU
  - Lots of file system ideas from Coda and InterMezzo
  - Recently (9/07) purchased by Sun
- Key design features
  - Direct client/storage communication
  - Proprietary object storage protocol (not SCSI/T10 OSD)
  - OSS (server) hosts multiple OST (targets)
  - Exotic network support (enhanced Sandia Portals stack)
  - RAID from block storage behind OST
  - RAID-0 striping across OST
  - OSS fail over if OST is dual-ported
  - Single metadata server with scripted fail over to backup
  - Distributed lock protocol among clients, OSS, and MDS



Lustre material from www.lustre.org and various talks

© 2008 Panasas. All Rights Reserved.

Lustre: Block Allocation

- Each OSS (object storage server) manages one or more (typically 2) OST (object storage target)
  - OSS network protocol is based on Sandia Portals network stack
  - OST is a modified EXT3 file system, which does block allocation
  - Currently porting to user-level ZFS for OST
- Files can be striped across >1 objects stored on different OST
  - No dynamic space management among OST (i.e., no object migration to balance capacity)
- Snapshots and Quota done independently in each OST

© 2008 Panasas. All Rights Reserved.

Lustre: Metadata

- MDS uses a modified EXT3 to store file system metadata
  - All the regular file system attributes, plus a storage map of the objects that contain its data
  - Single MDS with optional backup if the EXT3 is on shared disks
  - Clustered MDS is on the road map
- Distributed lock protocol among MDS, OSS, and Clients
  - "Intents" convey hints about the high-level file operations so the right locks can be taken and server round-trips avoided

© 2008 Panasas. All Rights Reserved.

Lustre: Caching

- MDS and OSS cache metadata, but not data
- Clients use the locking protocol to protect cached data and serialize access to data
  - Byte range locks for N-to-1 workload (i.e., shared output file)
  - Client cache shared with OS page/buffer cache

© 2008 Panasas. All Rights Reserved.

Lustre: Reliability

- Drive reliability comes from underlying RAID system
- OSS failover if the OST are on dual-ported drives
  - Large RAID systems sub-divided into several OST that are fronted by redundant OSS
- MDS failover if the EXT3 metadata file system is on dual-ported drives
  - Dedicated RAID array for this file system
- Crash recovery based on logs and transactions
  - MDS logs operation (e.g., file delete)
  - Later response from OSS cancels log entry
  - Some client crashes cause MDS log rollback

© 2008 Panasas. All Rights Reserved.

Other techniques

- Metadata database
  - Turns scalable metadata problem into scalable database problem
  - Database becomes a bottleneck if you're not careful
  - Loss of database makes storage contents meaningless
  - Lustre, many SAN filesystems, Google FS
- Weak cache consistency
  - NFS, AFS, etc.
  - Problematic when sharing is common
  - Most cluster applications have some degree of sharing, often write sharing
  - However, for some apps strong dirs + weak files may be enough

© 2008 Panasas. All Rights Reserved.

Real world issues

- Research filesystems are easy…
- What about?
  - Monitoring & troubleshooting
  - Backups
  - Snapshots
  - Capacity management – quotas, HSM, ILM
  - System expansion
  - Data migration

© 2008 Panasas. All Rights Reserved.

Themes

- Scalable clusters require scalable storage
  - Centralized/single anything eventually becomes a bottleneck
- File/object oriented storage API is superior to block oriented
  - Parallel, scalable block allocation
  - Block protocols have poor security & fencing support
  - Block layouts are cumbersome
- Reliability is important
  - Large systems will constantly have something that's broken
  - Tolerating failures is necessary to make forward progress

© 2008 Panasas. All Rights Reserved.

Questions?

welch@panasas.com

© 2008 Panasas. All Rights Reserved.