IBM Linux Technology Center

NFS-Ganesha
Why is it a better NFS server for Enterprise NAS?

Venkateswararao Jujjuri (**JV**)
File systems and Storage Architect
IBM Linux Technology center
jvrao@us.ibm.com | jujjuri@gmail.com

Linux Collaboration Summit 2014

Outline

- What is NFS-Ganesha
- Enterprise NAS
- Kernel vrs User-Space Server
- Clustering with NFS-Ganesha
- Failover/IP move and recovery
- New improvements
- Conclusions.

What is NFS-Ganesha?

- User-level implementation of NFS server
  - Supports V2, V3, V4, v4.1, v4.2
- Can manage huge meta-data and data caches
- Able to provide access to different sets of data
- Provision to exploit FS specific features.
- Can serve multiple types of File Systems at the same time.
- Can serve multiple protocols at the same time.
- Can act as a Proxy server and export a remote NFSv4 server.
- Cluster Manager agnostic.

- Small but growing community.
- Active participants - IBM, Panasas, Redhat, LinuxBox, CES

Enterprise NAS

- Reliable, Redundant and Fail-safe through clustered configuration.
- Serve structured and unstructured data over multiple protocols.
- Scalable in capacity and performance.
  - Flexible to Scale-up and/or Scale-out.
  - Capable of supporting large number of clients.
- Enterprise features – tiering, de-dup, multi-tenancy, multi-protocol etc.
- Flexible to run on various platforms and serve heterogeneous sets of data.
- Support complex security configurations.
- QoS – Quality of Service.

Kernel vs User Space Server

- Kernel Server need to make many up-calls
  - Mountd and exportfs info
  - ID mapping
  - GSSAPI
  - Client ID tracking (future?)
  - Statd interaction.
  - Cluster Related.

- One would question why do we need this in the kernel? (well it seemed like a good idea at that time)

Kernel vs User Space Server

- What is so great about user-space?
  - User Space is more flexible than kernel.
  - Easy restarts, failover, failback implementation.
  - System calls don't need to get in the way.
  - No up-calls to interact with user space services.
  - Clustering becomes natural and easy.
  - Targeted and aggressive caching capabilities.
  - Flexible and Plug-able FSAL
    - FS Specific features can be exploited.
  - Can support multi-Protocol with common DLM
  - Easy to achieve multi-tenancy
  - Easy to monitor and control resource consumption and even extend to enforcing QoS.
  - Manageability and debug-ability are major plus.

Kernel vs User Space Server

- No merits for kernel server? Yes there are.
  - Filehandles – Major advantage until recently; Now we have open-by-handle support.
  - Performance – User mode can be slow – but can be offset by Clustered FS, Aggressive Caching, Customized RPC, and aggressive threading/parallel execution
  - Ownership/permissions – workaround setfsuid per process. But others may need to do multiple system calls or special FS interface. VFS,Lustre,GPFS but CEPH, Gluster libraries can accept
  - Multiple syscalls may be needed to perform, write/getattr for WCC reasons.
  - No duplicate cache and Zero-Copy read/write is less complex.

Clustering with NFS-Ganesha

- Proposed **C**luster **M**anager **A**bstraction **L**ayer (CMAL) makes it cluster agnostic.
- Fail-over and IP move handling
- Cluster aware DRC
- DLM across cluster
- Cluster wide Grace period is manged.

Cluster Manager Abstraction Layer (CMAL)

- Provides an abstraction for cluster manager support.
- Manages intra-node communication among cluster nodes.
- Generic enough to implement many clustering interactions for cDRC, DLM etc features.
- Cluster manager agnostic. Easy to plug any cluster manager.
- Modeled after FSAL (File System Abstract Layer).
- CMAL layer would have function pointers based on the Cluster Manger.
- Can provide cluster communication between DS and MDS nodes in pNFS configuration.

Cluster DRC

- DRC helps the server to identify duplicate requests of non-idempotent operations and process accordingly.
- Identify a back-up node to backup the DRC or backup on all nodes of the cluster/central location
- Store and fetch DRC entries through CMAL.
- Interfaces
  - init_drc_cmal(server-ip): Initialize the CMAL interface. Set up backup node for the server-ip.
  - add_drc_entry(xid, entry, server-ip): Stores the DRC entry in the cluster.
  - retrieve_drc_entries(server-ip): Retrieve all the DRC entries for a particular sever node.
  - shutdown_drc_cmal(server-ip): Shutdown the CMAL interface for the given server-ip.

Failover/IP Move

- NLM makes it very complex but NFS-Ganesha architecture is up for the challenge. :)
- On the first lock/last unlock, Ganesha calls Cluster Manager provided interface to register(monitor)/unregister(unmonitor) the client-ip, server-ip pair.
- When the ip is moved (manual/failover), CM sends sm_notify to clients of the affected service-ip.
- CM generates events, release-ip and take-ip for corresponding server nodes, so that state shall be released from the source node, and acquired by the destination node.
- Depending on the lock granularity, corresponding locks/file systems or entire cluster should enter grace.

Distributed Lock manager

- The NFS-Ganesha architecture supports DLM.
  - DLM can be implemented in different ways.

- Allowing to manage cluster wide locks.

- Shared state management across protocols - NFS and CIFS and other.

- Ability to minimize the impact of failures.
  - Targeted Grace.

FSAL Enhancements and new features

- Dynamic Exports
- Delegations
- FSAL enhancements
  - ACL support.
  - Open upgrade support.
  - Working around POSIX semantics.
  - User libraries can be plugged easily.
    - ProtecTIER, GlusterFS.
  - Stackable FSALs.
  - PseudoFS as first class FSAL.
- LTTng Integration.
- RDMA support through libmooshika.

NFS-Ganesha links

- NFS-Ganesha is available under the terms of the LGPLv3 license.
- NFS-Ganesha Project homepage on github
  - https://github.com/nfs-ganesha/nfs-ganesha/wiki
- Github:
  - https://github.com/nfs-ganesha/nfs-ganesh
- Download page
  - http://sourceforge.net/projects/nfs-ganesha/files

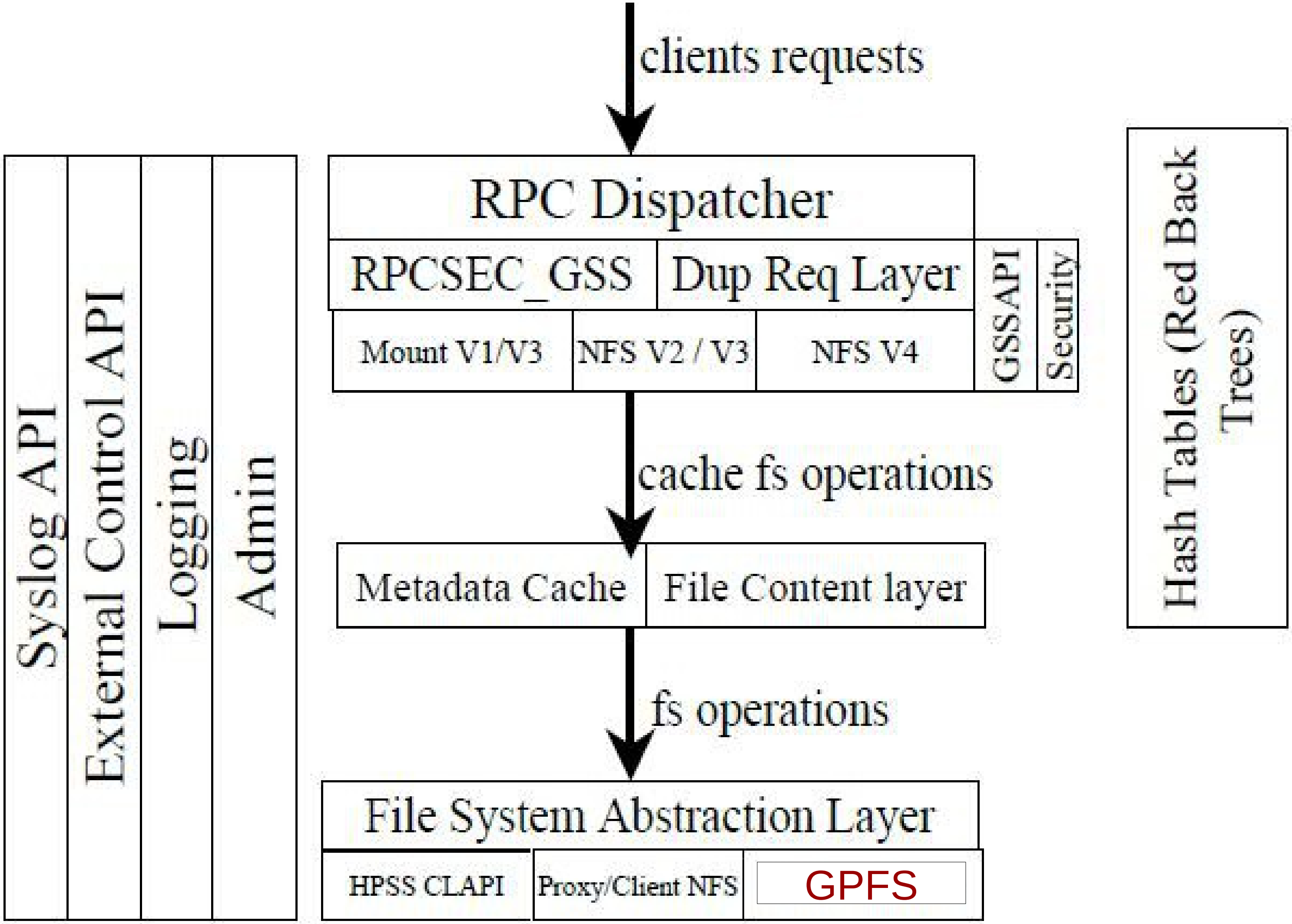
- Mailing lists
  - nfs-ganesha-devel@lists.sourceforge.net
  - nfs-ganesha-support@lists.sourceforge.net
  - nfs-ganesha-announce@lists.sourceforge.net

Legal Statements

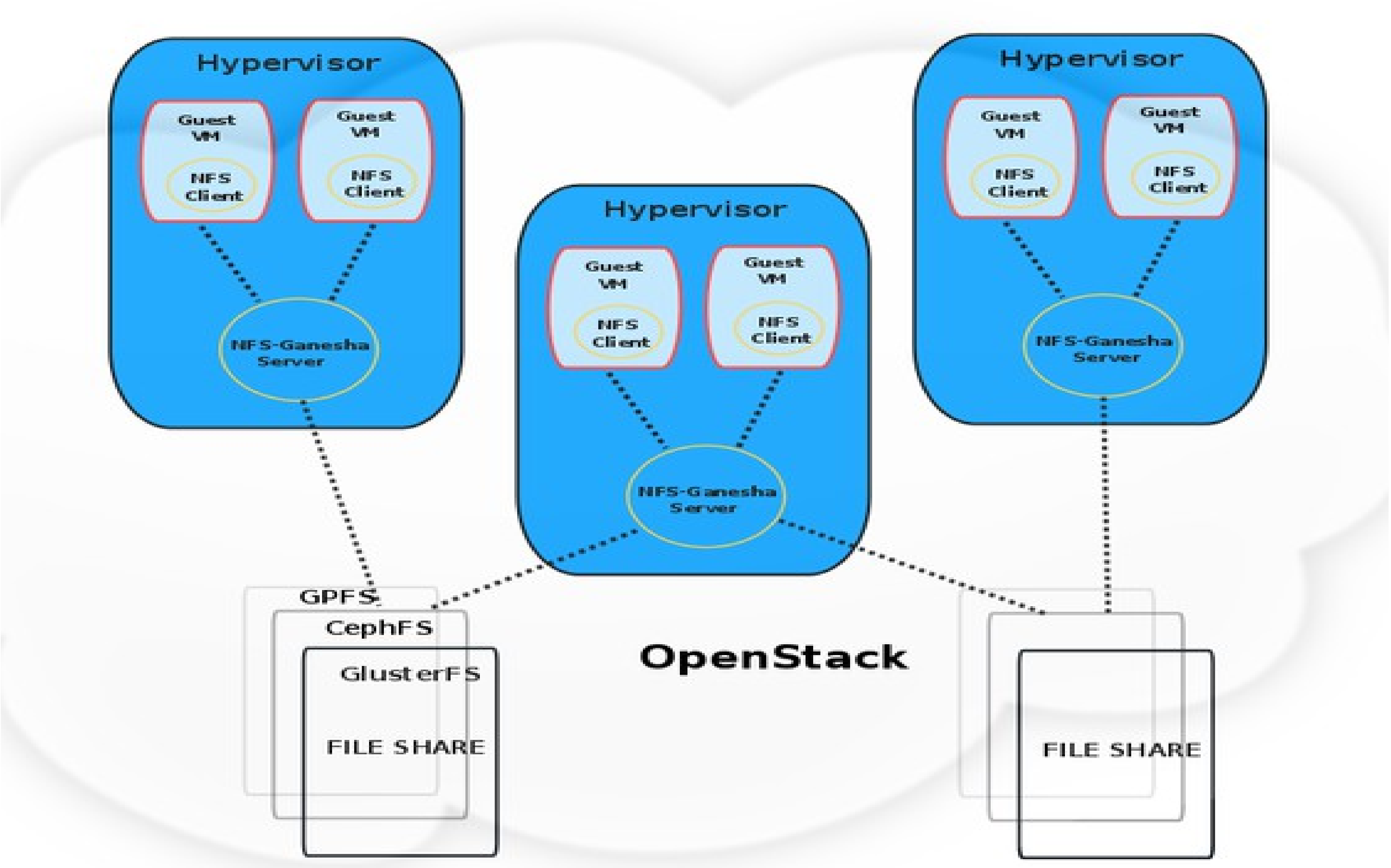
- This work represents the view of the author and does not necessarily represent the view of IBM.
- IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.
- UNIX is a registered trademark of The Open Group in the United States and other countries .
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Other company, product, and service names may be trademarks or service marks of others
- CONTENTS are "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you. This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Author/IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

BACKUP

Architecture

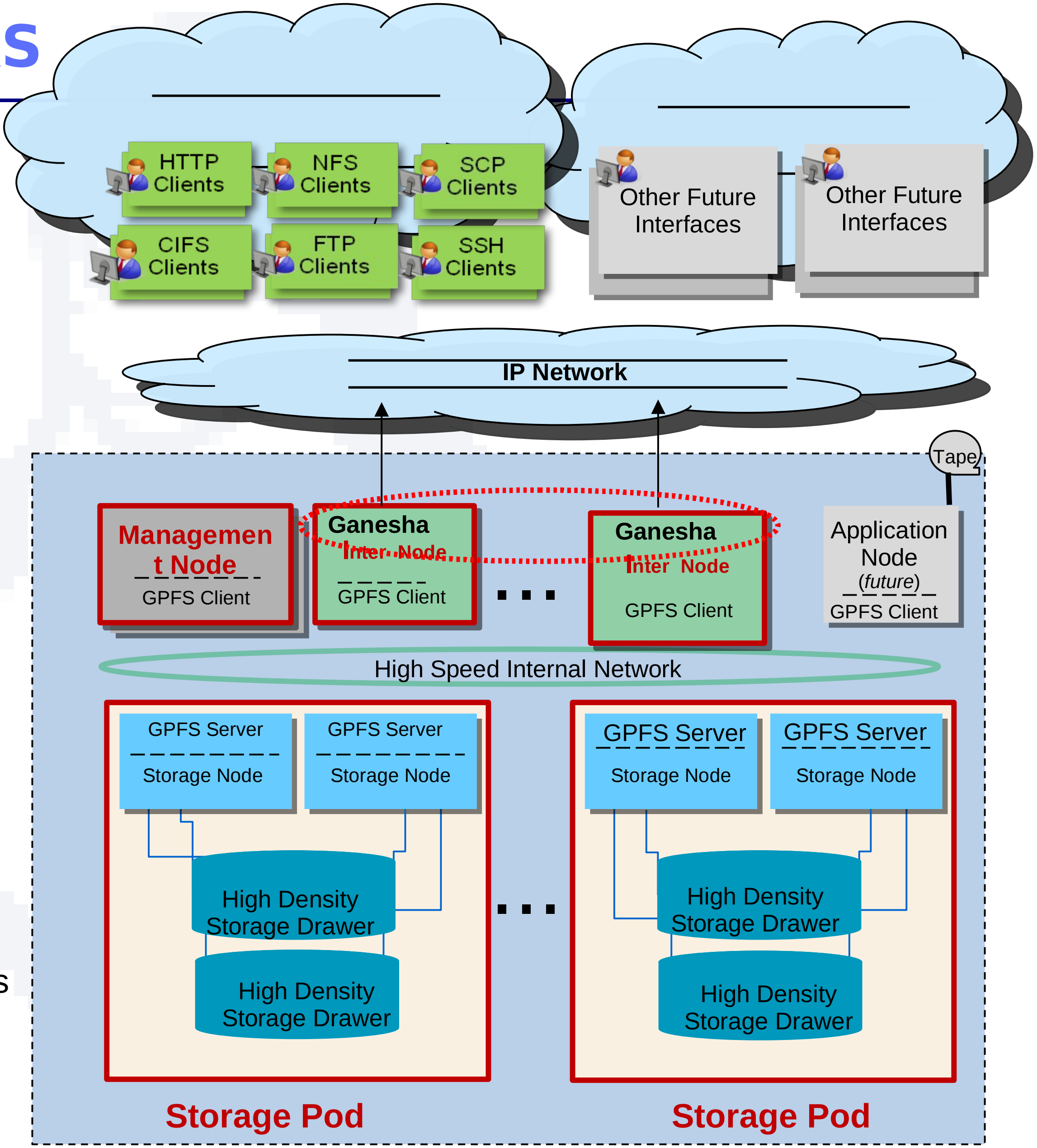


Manila Open-Stack using NFS-Ganesha (Gateway mediated)



Use Cases - SoNAS

High level design of Cloud Storage infrastructure

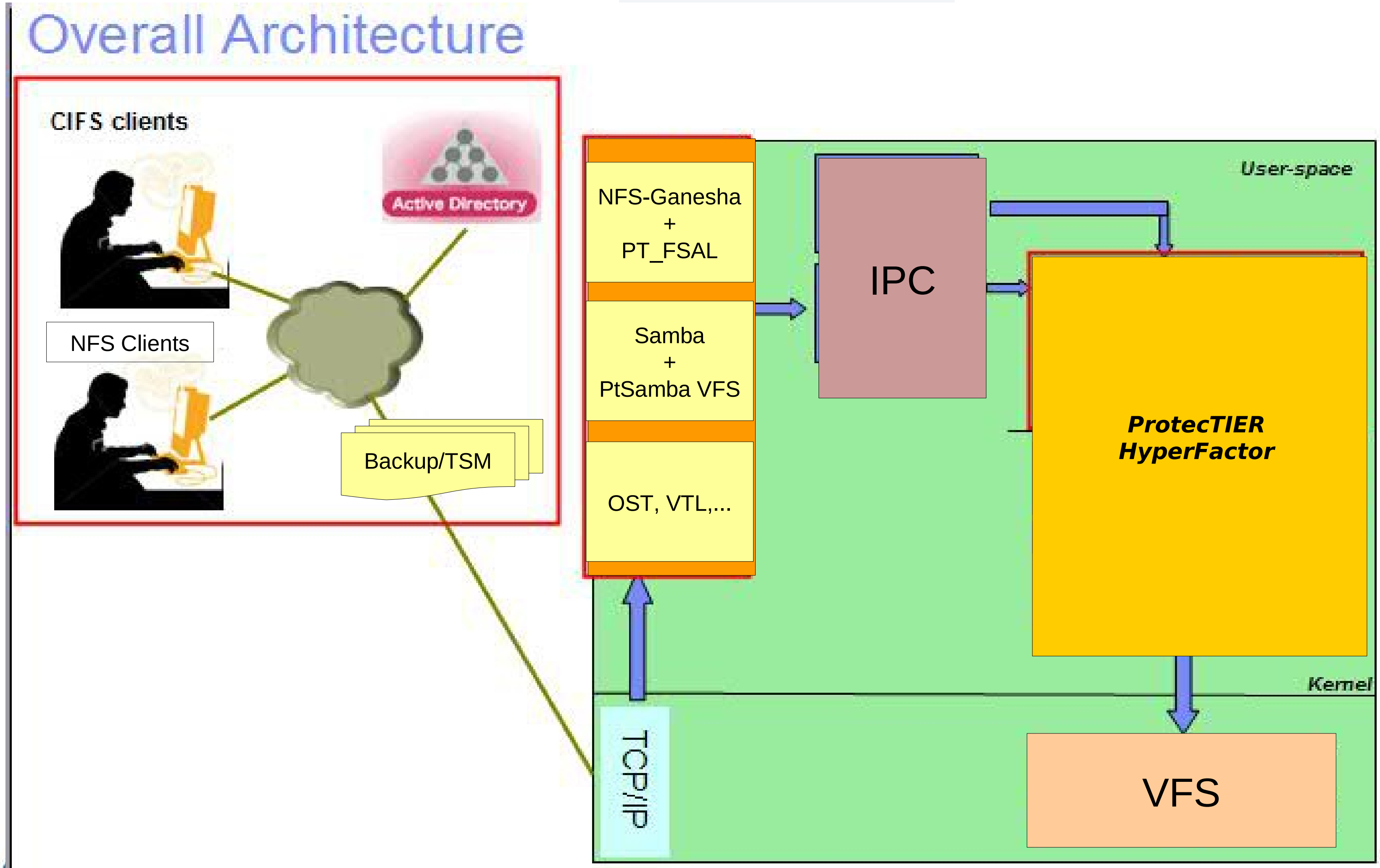


Simple: Only 3 basic parts:

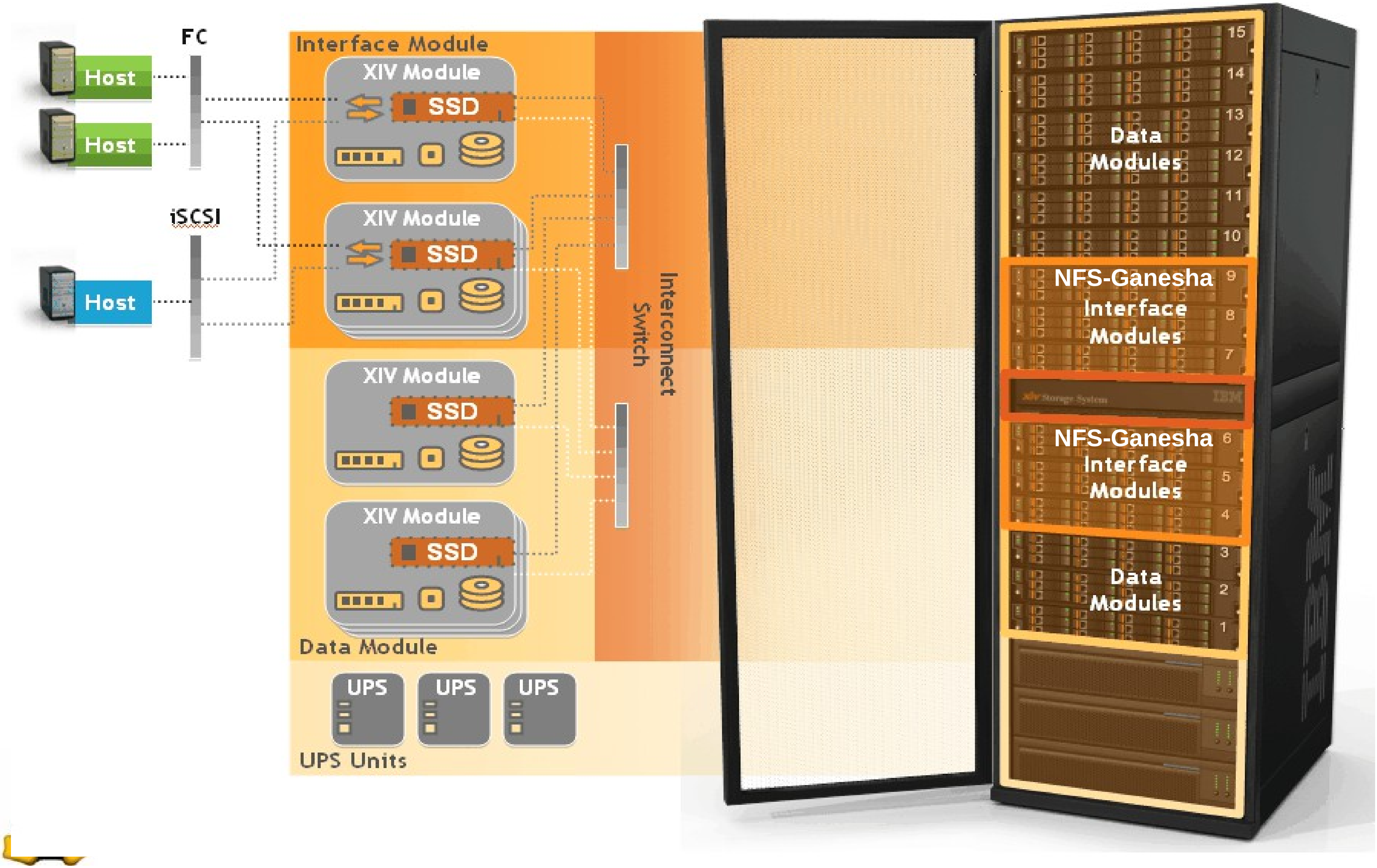
- ✓ **Interface Nodes**
- ✓ **Management Nodes**
- ✓ **Storage Pods**

'Lego-like' modular design allows online independent scaling of I/O capacity and Storage capacity

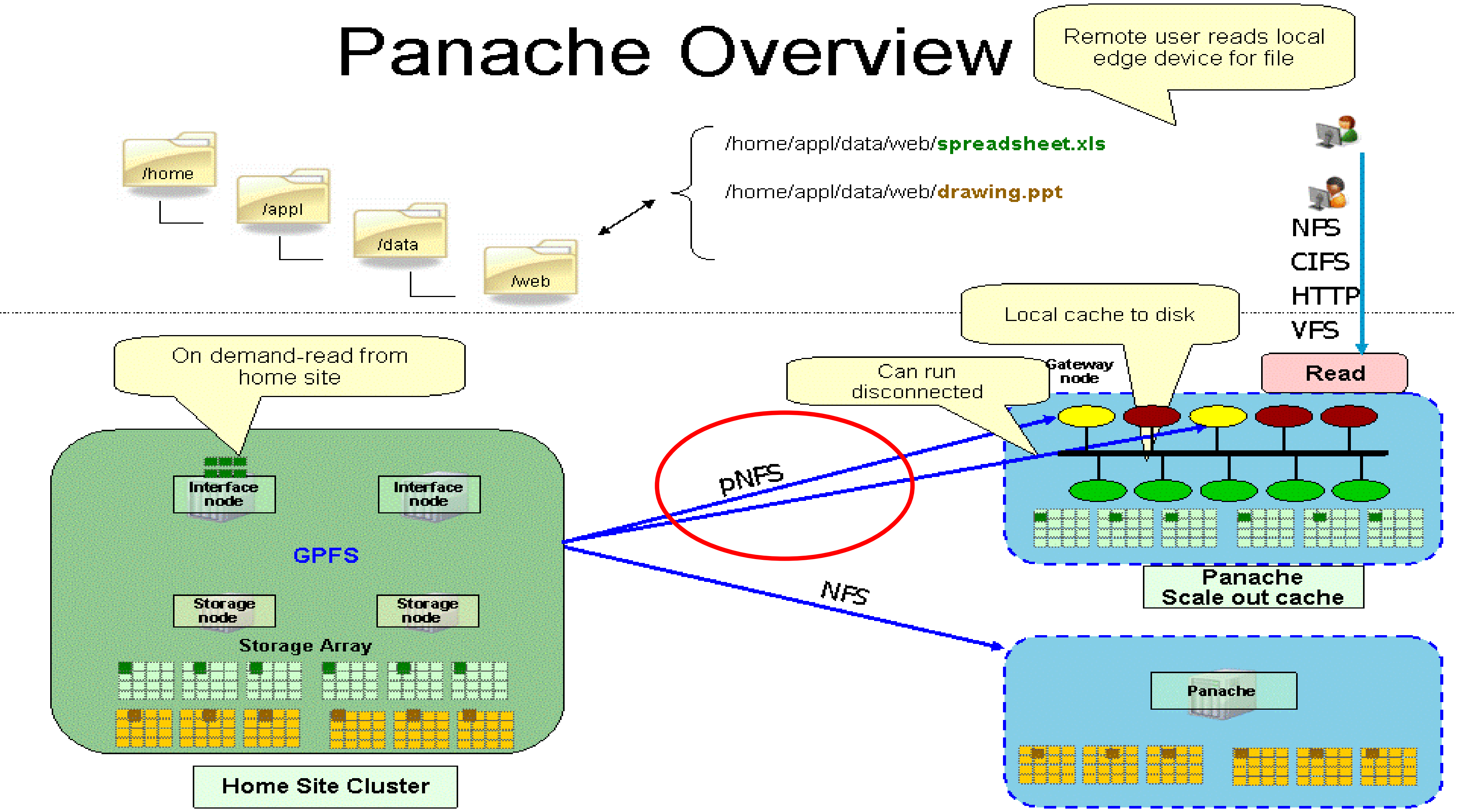
Use Cases - ProtecTIER



XIV Architecture

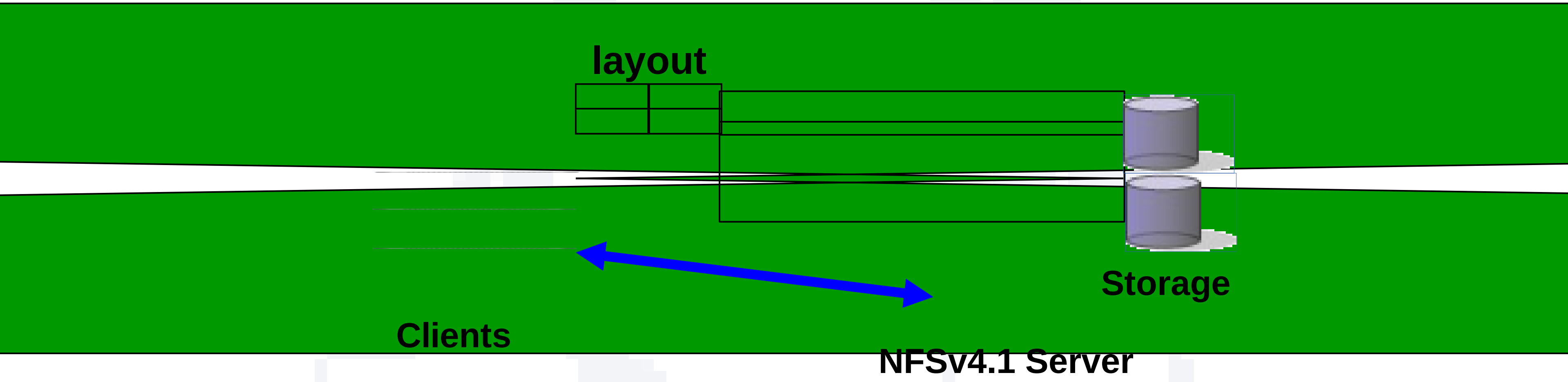


Panache UseCase



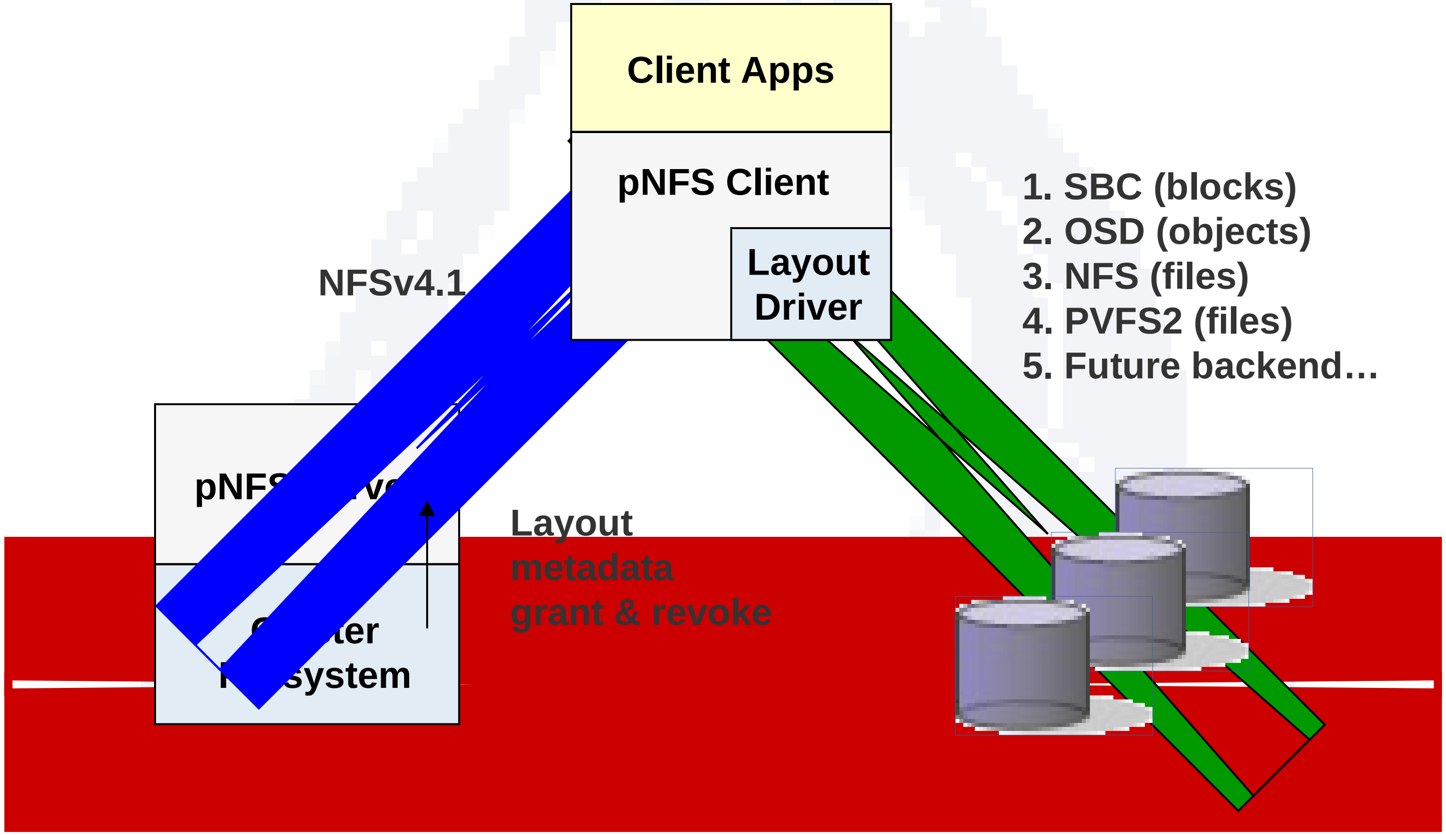
pNFS Layouts

- 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



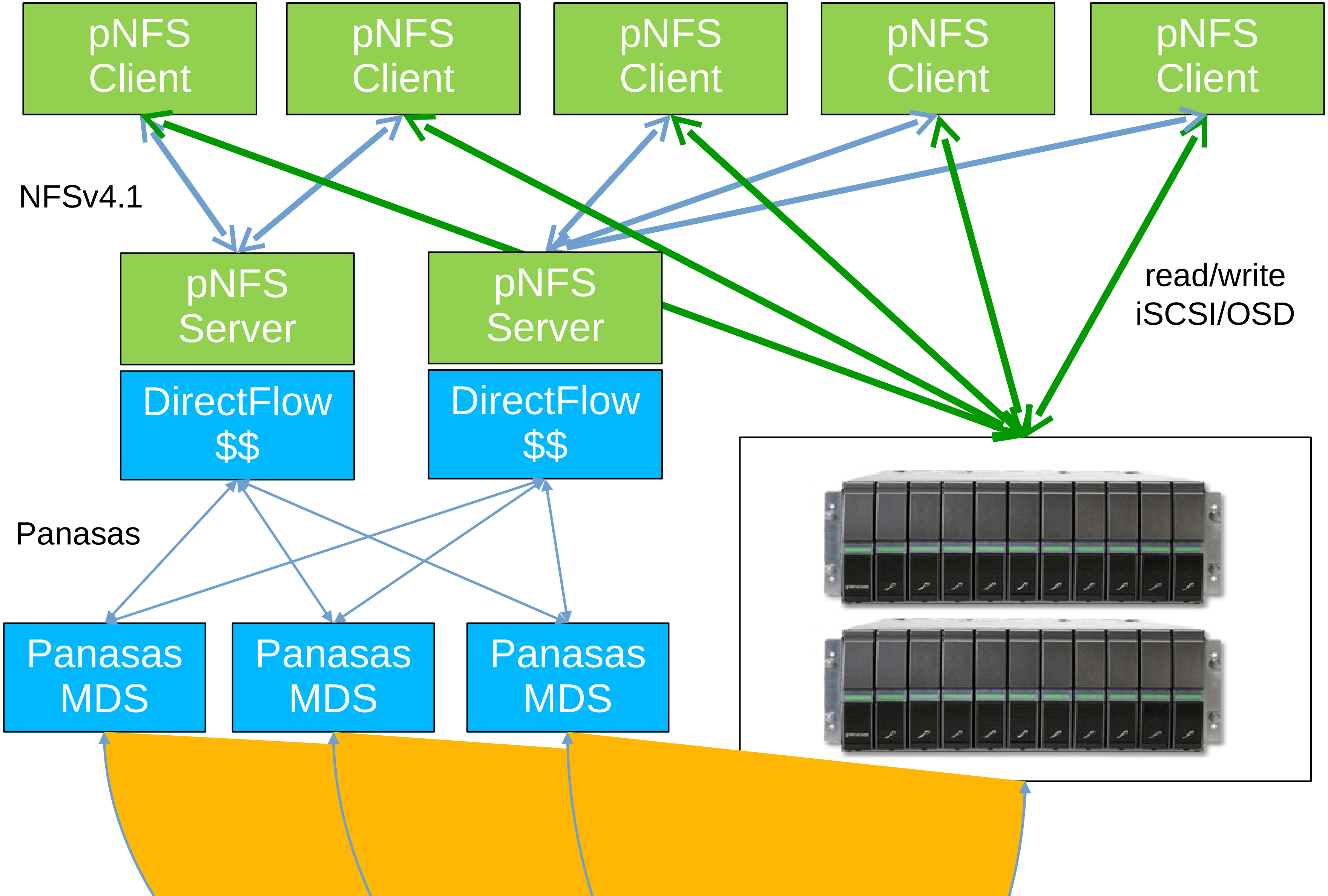
pNFS Client

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

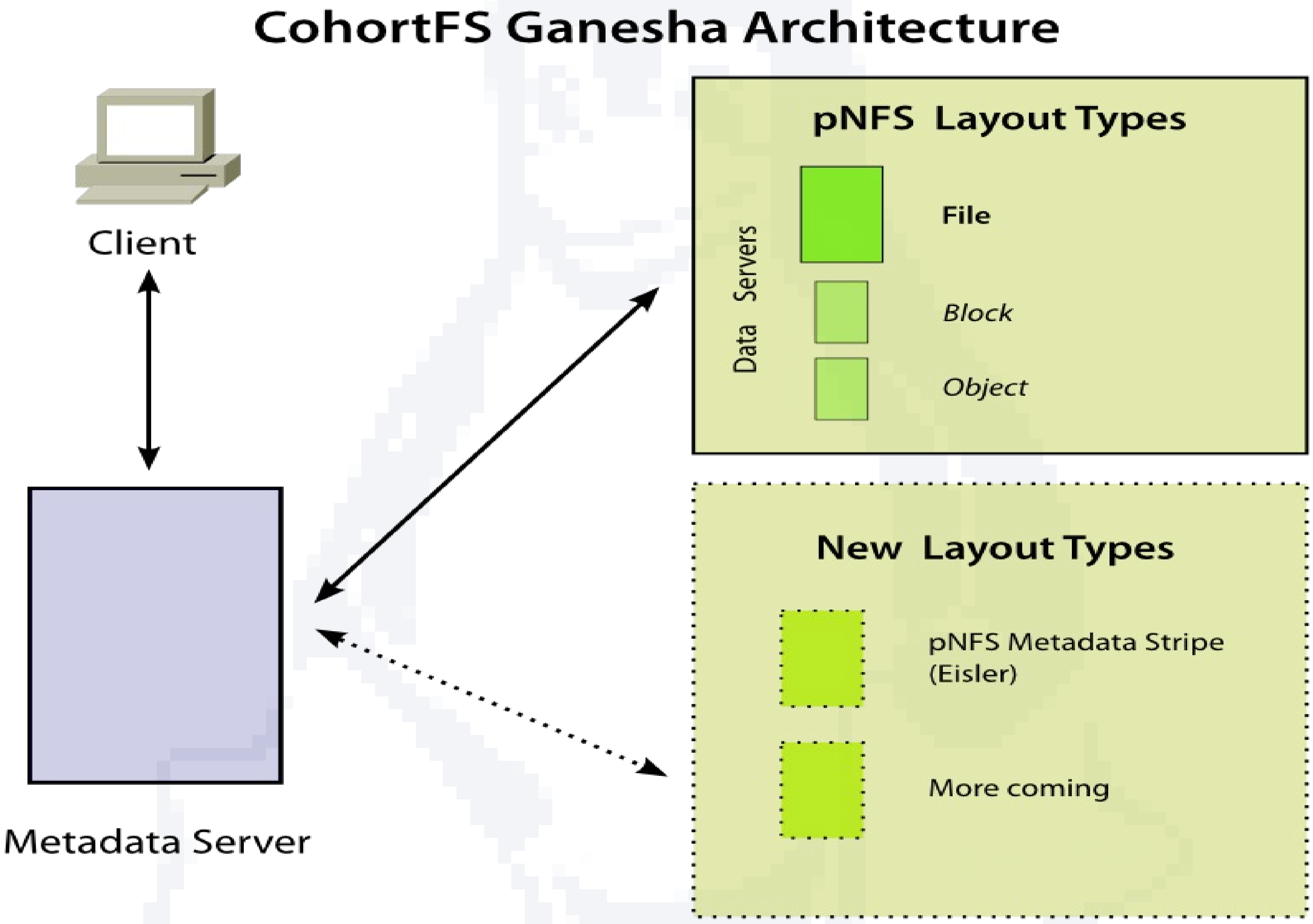


1. SBC (blocks)
2. OSD (objects)
3. NFS (files)
4. PVFS2 (files)
5. Future backend…

Scaling pNFS metadata service



UseCase - CohortFS(LinuxBox)


CohortFS Ganesha Architecture
Client
Metadata Server
pNFS Layout Types
Data Servers
File
Block
Object
New Layout Types
pNFS Metadata Stripe
(Eisler)
More coming