Architecture of a Next-Generation Parallel File System

Agenda

- Introduction
- OrangeFS features
- OrangeFS futures
- Q&A

An introduction

Why Parallel File Systems?

- HPC and Big Data applications increasingly rely on I/O subsystems
  - Large input datasets, checkpointing, visualization
- Programmers need interfaces that match their problem
  - Multidimensional arrays, typed data, portable formats
- Two issues to be resolved by I/O system
  - Performance requirements (concurrent access to HW)
  - Gap between app. abstractions and HW abstractions
- Software is required to address both of these problems

What is OrangeFS?

- OrangeFS is a next generation Parallel File System
  - Based on PVFS
  - Distributes file data across multiple file servers leveraging any block level file system.
  - Distributed Meta Data across all servers using Berkeley DB
  - Supports simultaneous access by multiple clients, including Windows
  - Works w/ standard kernel releases and does not require custom kernel patches
  - Easy to install and maintain

PVFS to OrangeFS

1994-2004

Design and Development at
CU Dr. Ligon + ANL (CU Graduates)

2004-2010 **PVFS**

Primary Maint & Development
ANL (CU Graduates) + Community

2007-2010

OrangeFS PVFS Branch
initial development at CU + OB

SC10 (fall 2010) **PVFS**

Announced to community and is now
Mainline of PVFS development
Commercial Grade Services available

2012 **OrangeFS** 2.8.5, 2.8.6, 2.9.0

Toward Exascale

Future **OrangeFS NEXT** **PxFS**

dice PROGRAM
A Test & Research Environment for Innovation

Parallel File System Survey Report

9 December 2010

File systems used at Data Center	
CIFS/SMB	14.8%
CXFS	29.6%
GPFS	48.1%
Lustre	59.3%
NFS	74.1%
PanFS	29.6%
pNFS	11.1%
PVFS2	18.5%
Redhat GFS	11.1%
StorNext	7.4%
XFS	25.9%
ZFS	7.4%
Other	25.9%

Original PVFS Design Goals

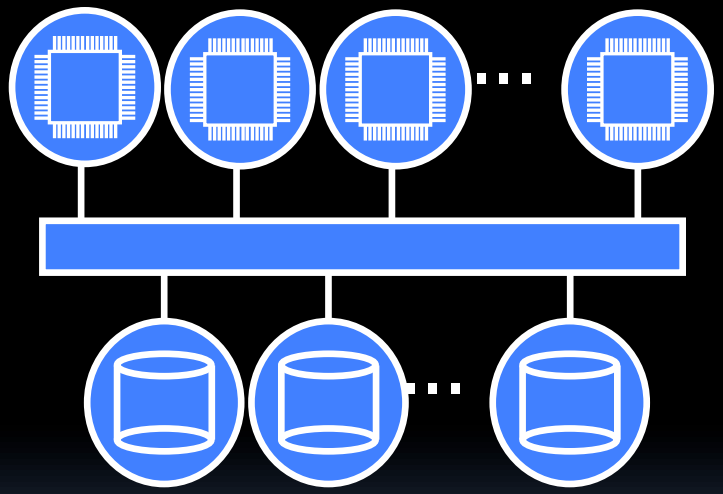
- Scalable
  - Configurable file striping
  - Non-contiguous I/O patterns
  - Eliminates bottlenecks in I/O path
  - Does not need locks for metadata ops
  - Does not need locks for non-conflicting applications
- Usability
  - Very easy to install, small VFS kernel driver
  - Modular design for disk, network, etc
  - Easy to extend

OrangeFS Philosophy

- Focus on a Broader Set of Applications
- Customer & Community Focused
- Embrace Research
- Completely Open Source
- Commercially Viable

Features

Target Architecture



Clients running applications
(1000s-1,000,000s)

Storage or System Network

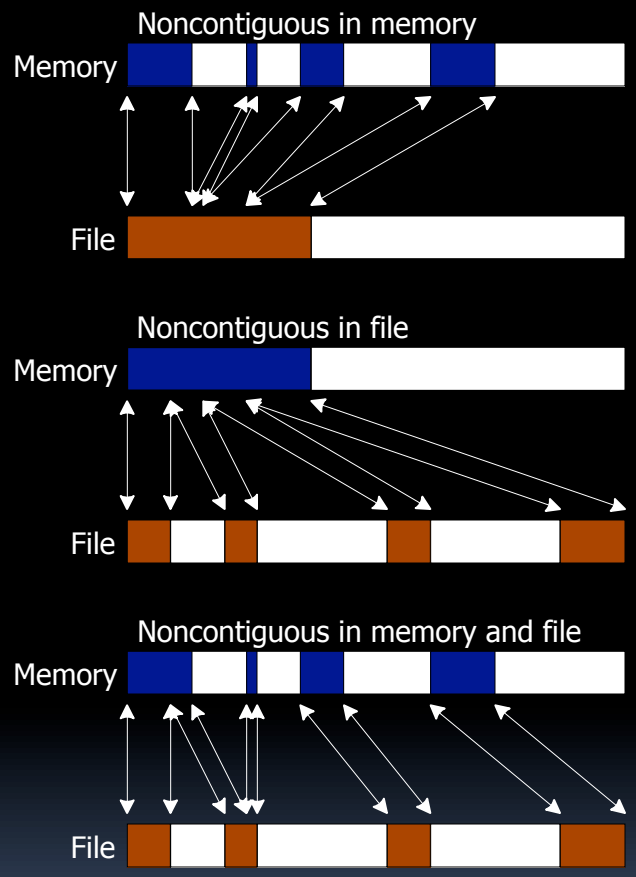
I/O devices or servers
(100s-10,000s)

System Architecture

- OrangeFS servers manage objects
  - Objects map to a specific server
  - Objects store data or metadata
  - Request protocol specifies operations on one or more objects
- OrangeFS object implementation
  - Berkeley DB for indexing key/value data
  - Local file system for stream of bytes

Non-contiguous I/O

- Noncontiguous I/O operations are common in computational science applications
- Most PFSs available today implement a POSIX-like interface (open, write, close)
- POSIX noncontiguous support is poor:
  - readv/writev only good for noncontiguous in memory
  - POSIX listio requires matching sizes in memory and file
- Better interfaces allow for better scalability



Semantics

- As we approach Exascale the reality of the limits of Sequential Consistency become more questionable. They are:
  - Expensive to implement for performance and scalability
  - Not needed if applications are well behaved
- OrangeFS uses a scalable lockless consistency model
  - Indistinguishable from SC for many programs
  - Provides much better performance/scalability
  - If the application requires locks, OrangeFS supports Distributed Lock Managers or application level locking, ex. Webdav interface for OrangeFS implements in metadata.

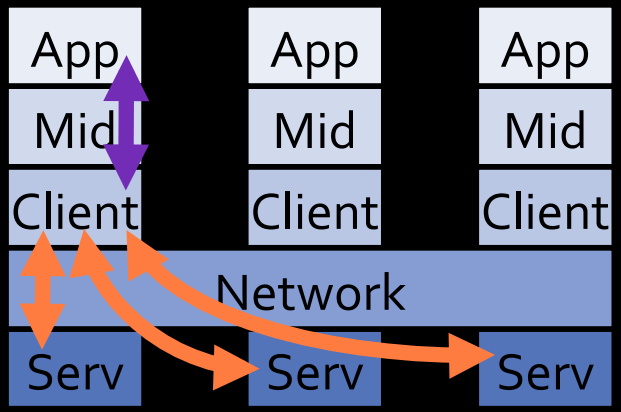
Recent Focus Areas

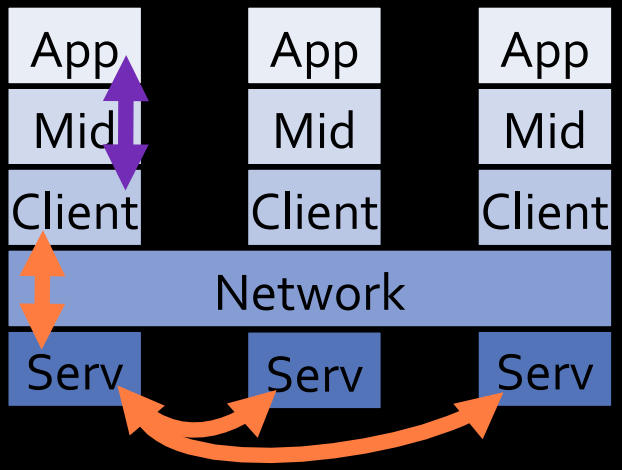
- Metadata Access Performance
- Reliability At Scale
- Security
- Diverse Access Methods
- Configurable Features
  - Avoid performance penalty for unused features

Recently Added to OrangeFS

- In 2.8.3
  - Server-to-Server Communication
  - SSD Metadata Storage
  - Replicate on Immutable
- 2.8.4, 2.8.5 (fixes, support for newer kernels)
- Windows Client
- In 2.9.0 (1st half 2012)
  - Distributed Metadata for Directory Entries
  - Capability-Based Access Control
  - Direct Access Libraries
    - preload library for applications
    - Including Optional Client Cache

Server to Server Communications





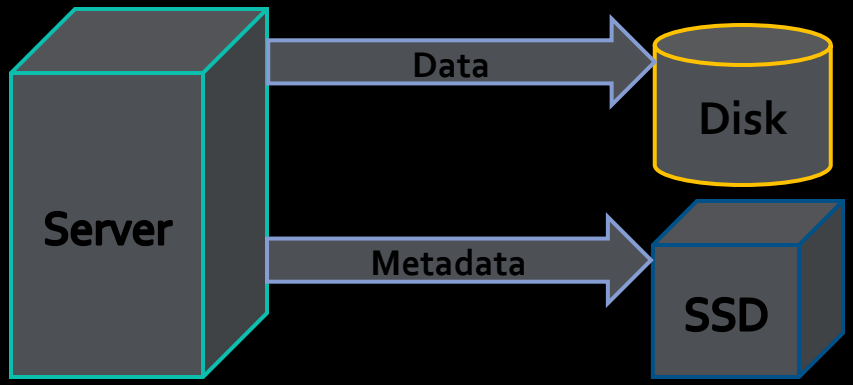
Traditional Metadata Operation

Create request causes client to communicate with all servers O(p)

Scalable Metadata Operation

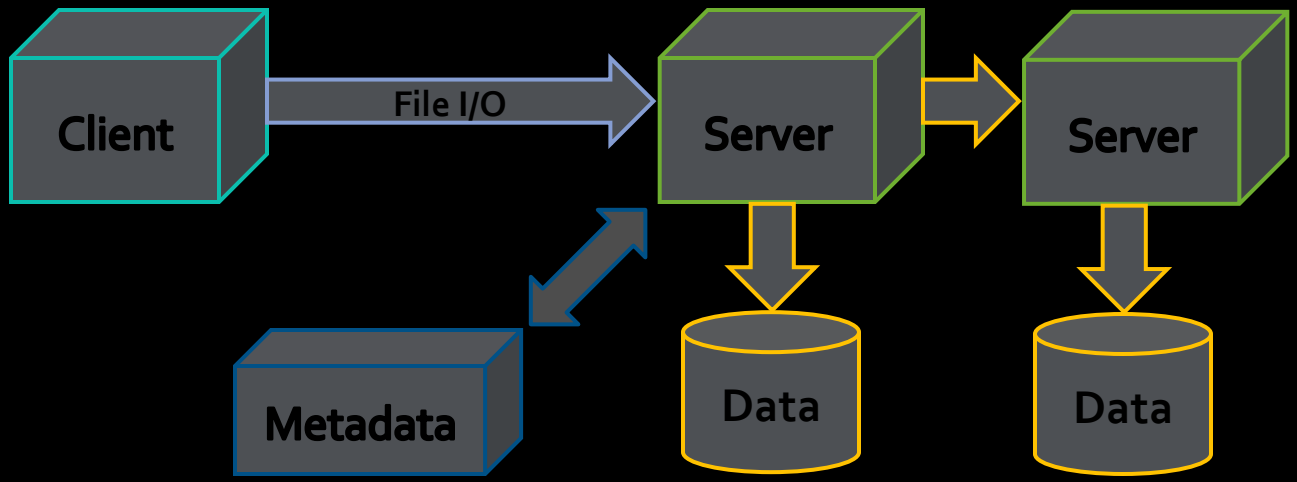
Create request communicates with a single server which in turn communicates with other servers using a tree-based protocol O(log p)

SSD Metadata Storage



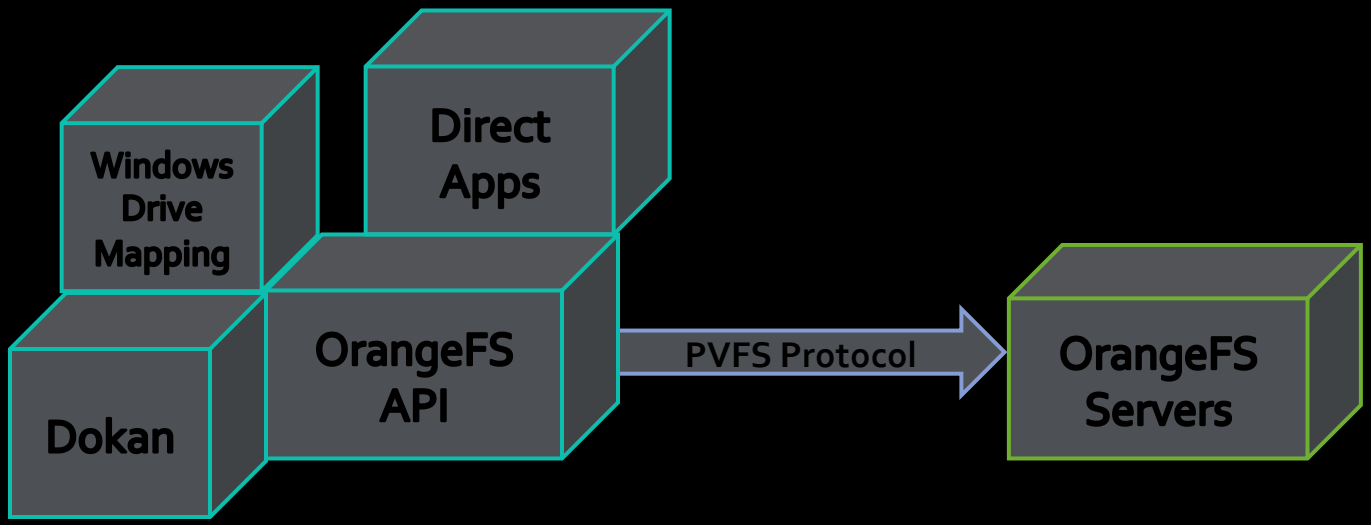
- Writing metadata to SSD
  - Improves Performance
  - Maintains Reliability

Replicate On Immutable



- First Step in Replication Roadmap
- Replicate data to provide resiliency
  - Initially replicate on Immutable
  - Client read fails over to replicated file if primary is unavailable

Windows Client

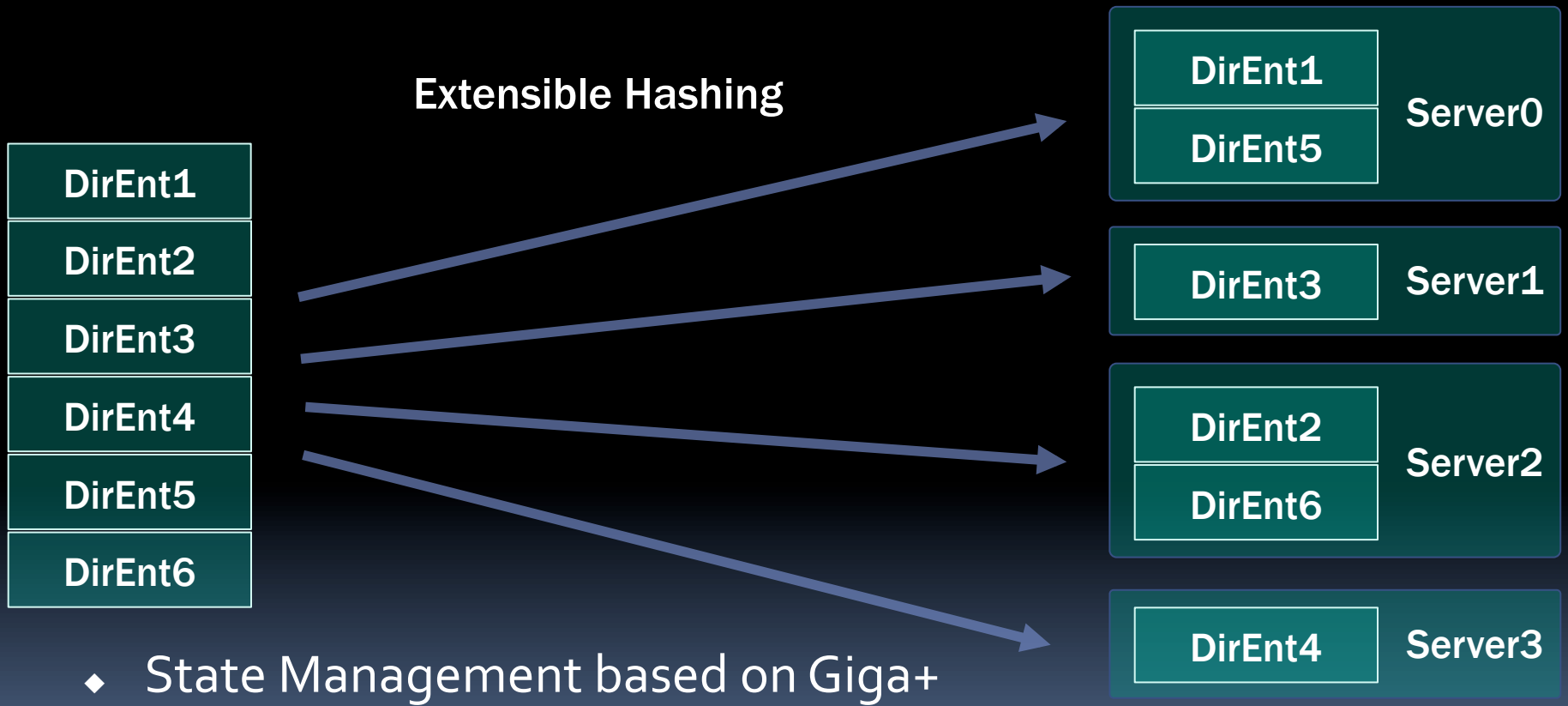


- Supports Windows 32/64 bit
- Server 2008, R2, Vista, 7

Coming in 2.9.0

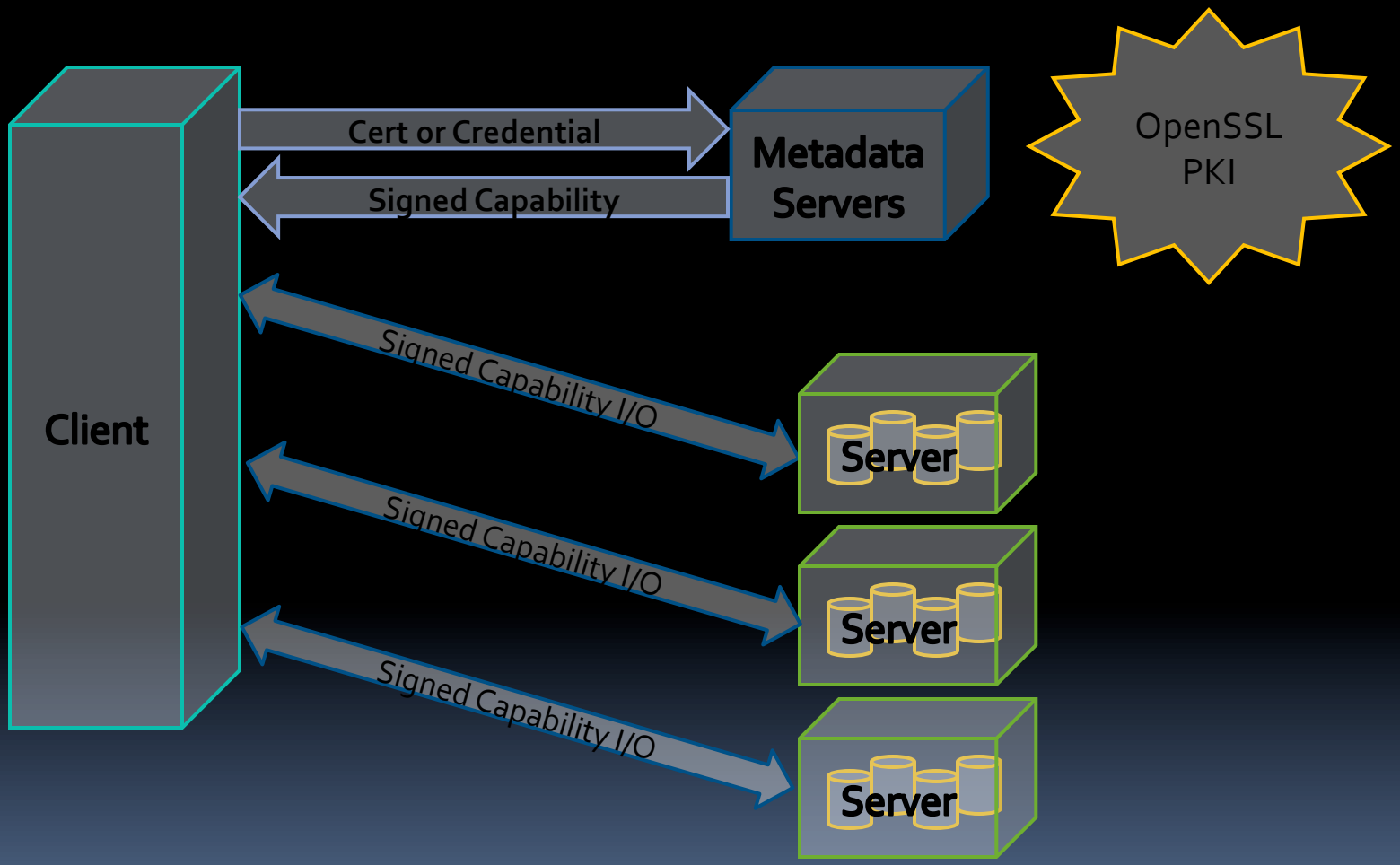
- In 2.9.0 (1st half 2012)
  - Distributed Metadata for Directory Entries
  - Capability-Based Access Control
  - Direct Access Libraries (preload library for applications)

Distributed Directory Metadata



- State Management based on Giga+
  - Garth Gibson, CMU
- Improves access times for directories with a very large number of entries

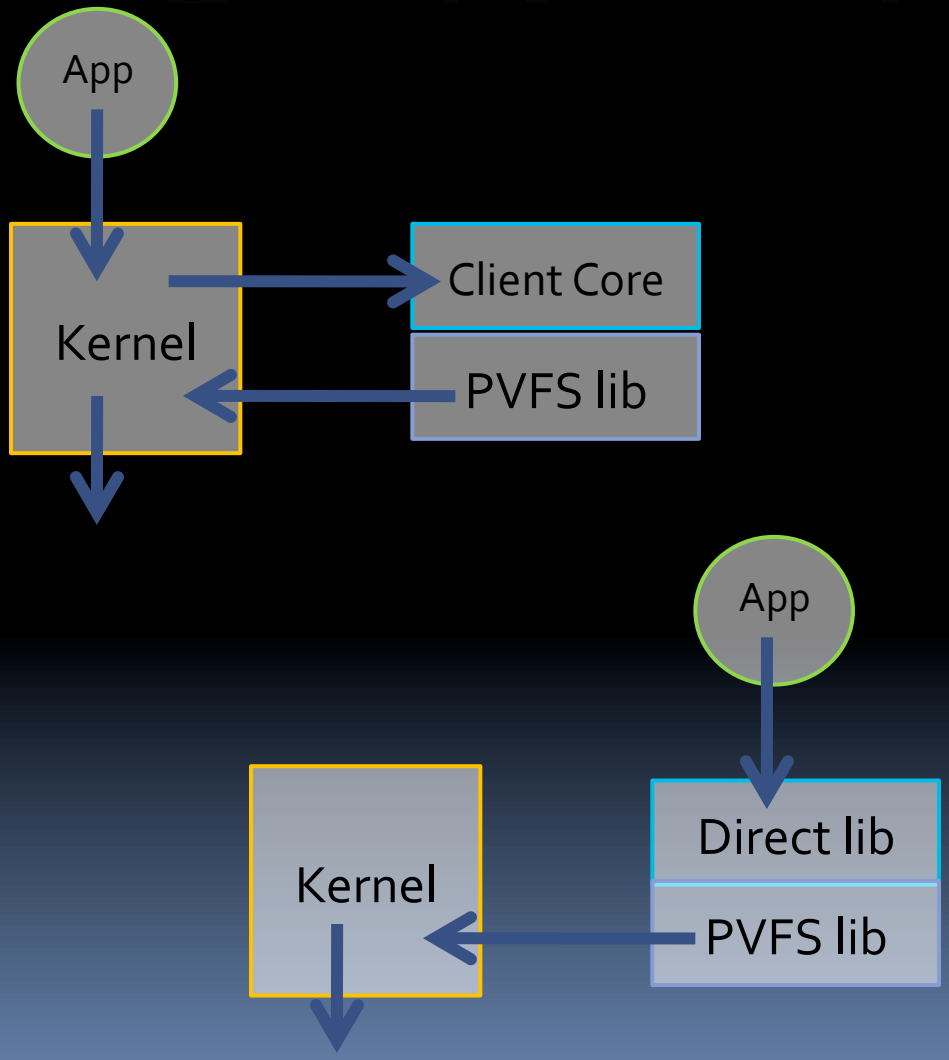
Capability Based Security



Extended Interfaces

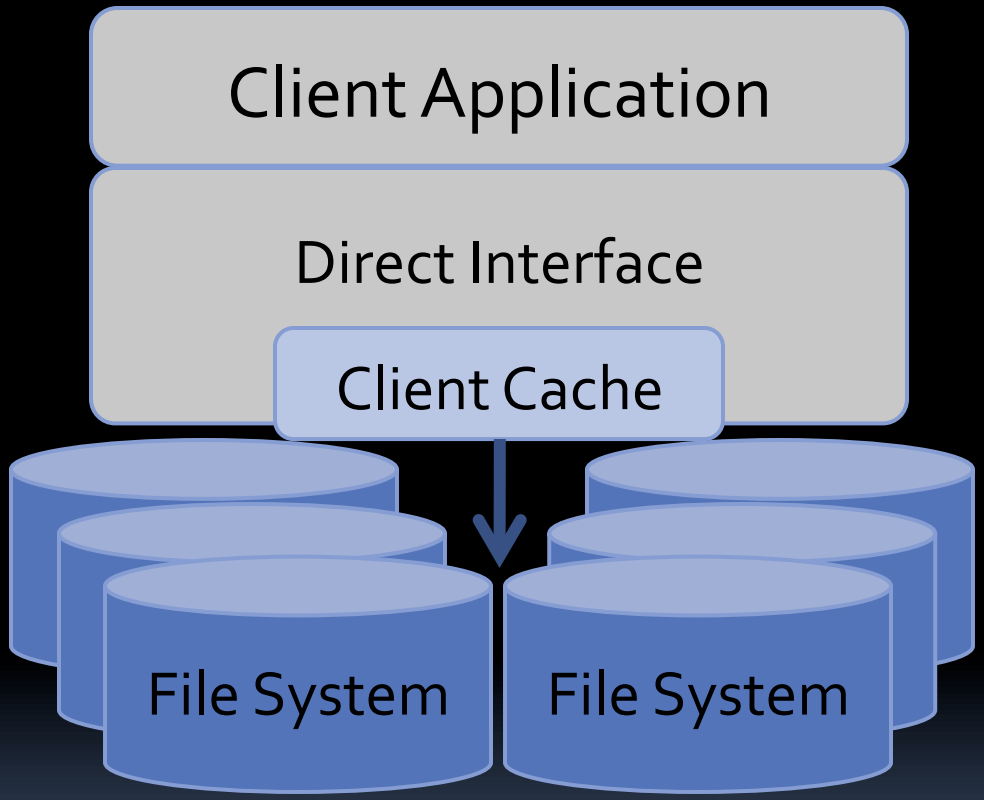
- Windows client (available now)
- FUSE (available now, improved mac support in 2.8.6)
  - Wider range of clients
  - Better stability using broader community code
- Web Package (Apache Modules) (2.8.6)
  - WebDAV Client
  - S3
  - REST Admin (DojoToolkit UI)
- Direct Access Libraries (2.9.0)
  - Better performance by bypassing kernel
  - Access OrangeFS and other files
  - Interface extensions
  - Easy to preload or link to applications

Direct Access Interface



- Implements:
  - POSIX system calls
  - Stdio library calls
- Parallel extensions
  - Noncontiguous I/O
  - Non-blocking I/O
- MPI-IO library

Direct Interface Client Caching



- Direct Interface enables Multi-Process Coherent Client Caching for a single client

Background Parallel Processing Infrastructure

- Modular infrastructure to easily build background parallel processes for the file system

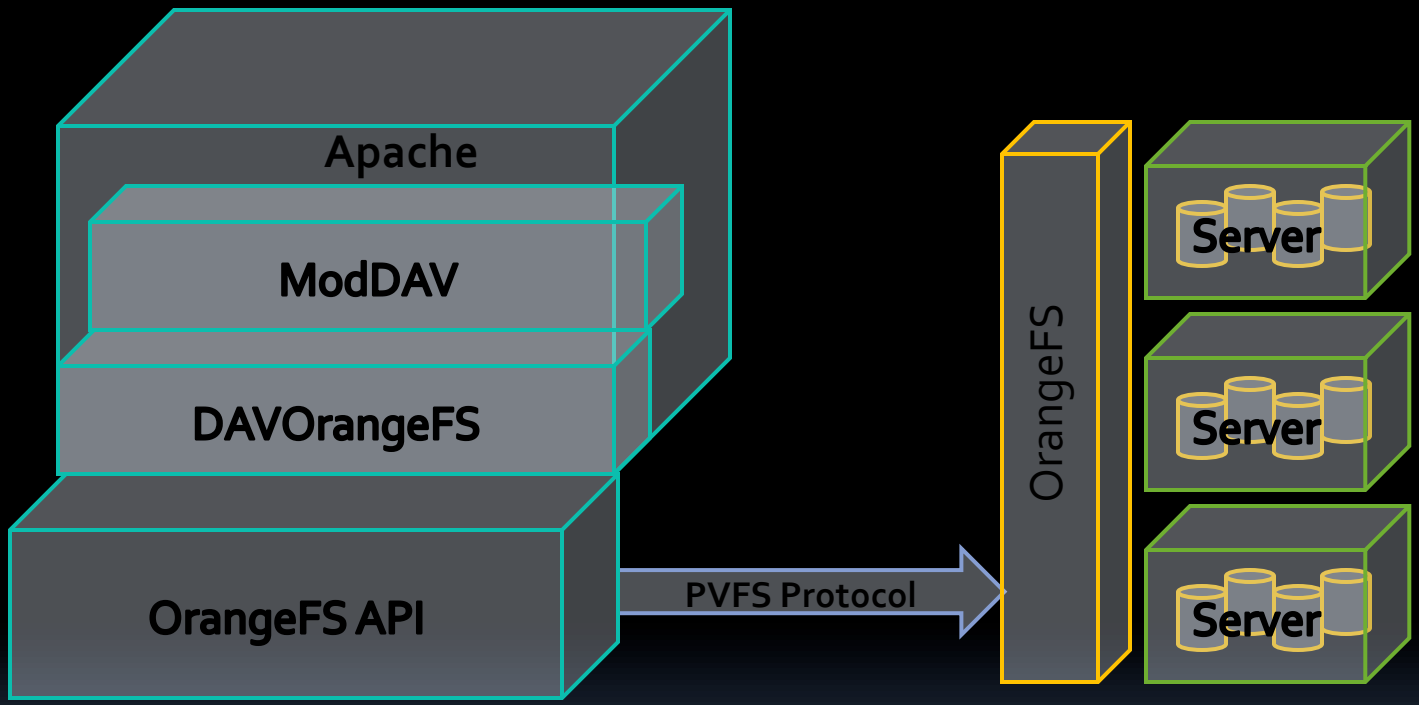
  Used for:

  - Gathering Stats for Monitoring
  - Usage Calculation (can be leveraged for Directory Space Restrictions, chargebacks)
  - Background safe FSCK processing (can mark bad items in MD)
  - Background Checksum comparisons
  - Etc…

Web Package

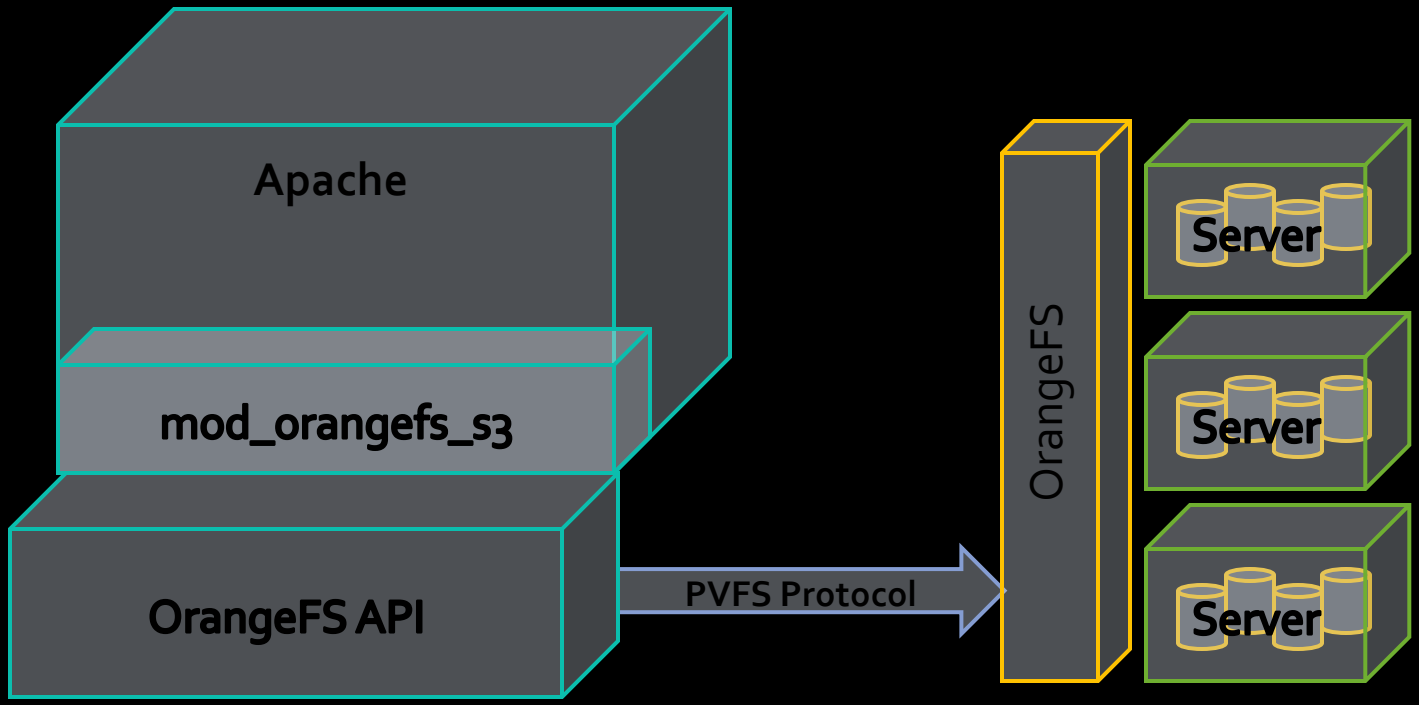
- WebDAV (2.8.x)
- S3 (2.8.x)
- REST Admin Interface (2.9.x)
- DojoToolkit Admin UI (2.9.x)

WebDAV



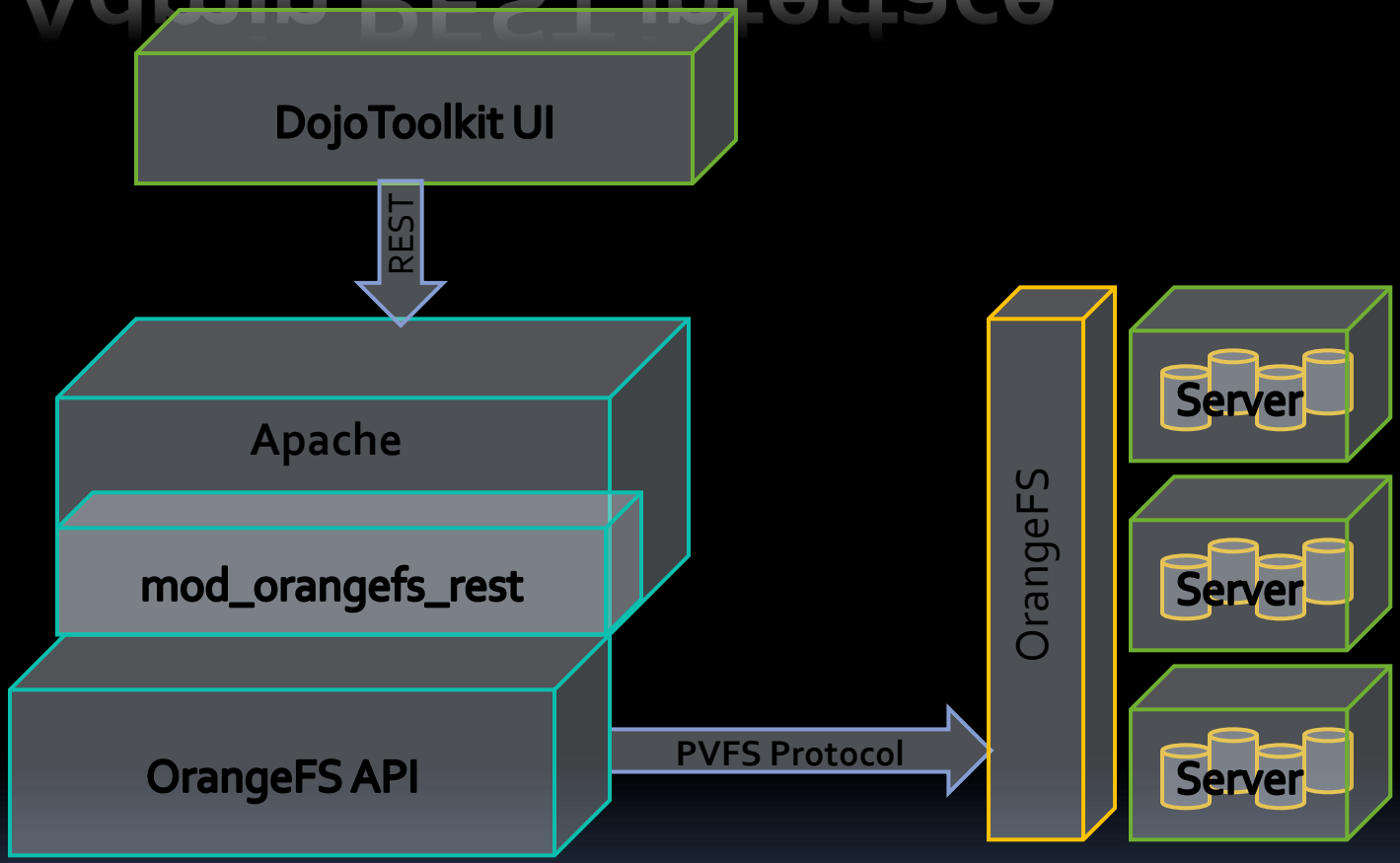
- Supports DAV protocol and tested with the Litmus DAV test suite
- Supports DAV cooperative locking in metadata

S3



- Tested using s3cmd client

Admin REST interface



- Tested using the DojoToolkit UI

Futures

OrangeFS NEXT

- 4 Foundational Elements of NEXT
  - File Handles –> 128bit UUID
  - Server Location and SID (Server Identifier) Management
  - Policy Based Configurable Replication, Migration and Hierarchical Storage
  - Attribute Based Metadata Search

Handles -> UUIDs

- An OID (object identifier) is a 128-bit UUID that is unique to the data-space
- An SID (server identifier) is a 128-bit UUID that is unique to each server.
- No more than one copy of a given data-space can exist on any server
- The (OID, SID) tuple is unique within the file system.
- (OID, SID1), (OID, SID2), (OID, SID3) are copies of the object identifier on different servers.

Server Location / SID Mgt

- In an Exascale environment with the potential for thousands of I/O servers, it will no longer be feasible for each server to know about all other servers.
- Servers will discover a subset of their neighbors at startup (or may be cached from previous startups). Similar to DNS domains.
- Servers will learn about unknown servers on an as needed basis and cache them. Similar to DNS query mechanisms (root servers, authoritative domain servers).

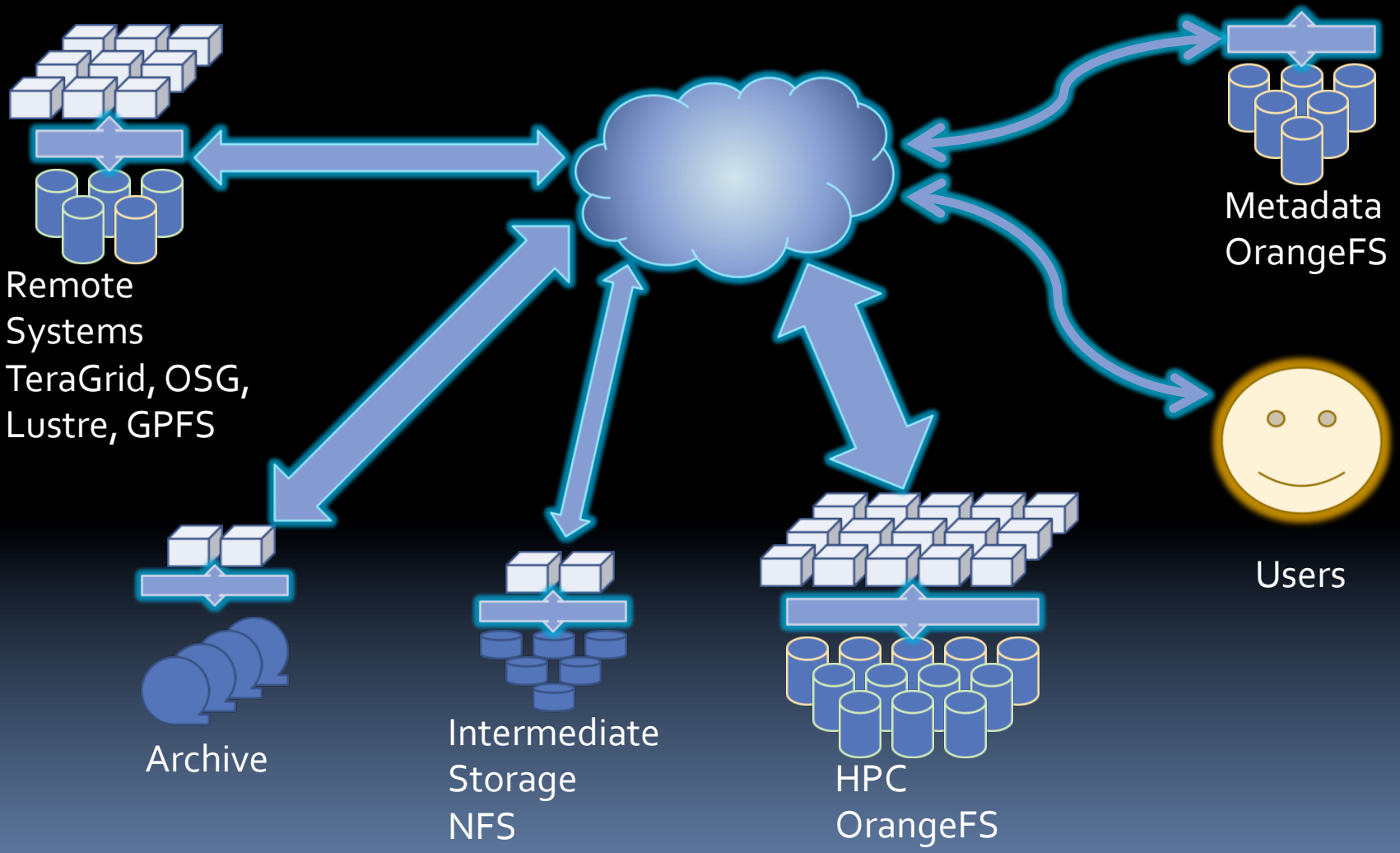
Replication / Redundancy

- Redundant Metadata
  - Easier recovery after a crash
  - Redundant objects from root directory down
  - Configurable
- Fully Redundant Data
  - Experiments with "forked flow" show small overhead
  - Configurable
    - Number of duplicates (O .. N)
    - Update mode (continuous, on close, on immutable, none)
- Emphasis on continuous operation

Migration

- Migrate objects between servers
  - De-populate a server going out of service
  - Populate a newly activated server
- Based on redundancy technology
  - Make a copy, then remove the old one
- Hierarchical storage
  - Use existing metadata services

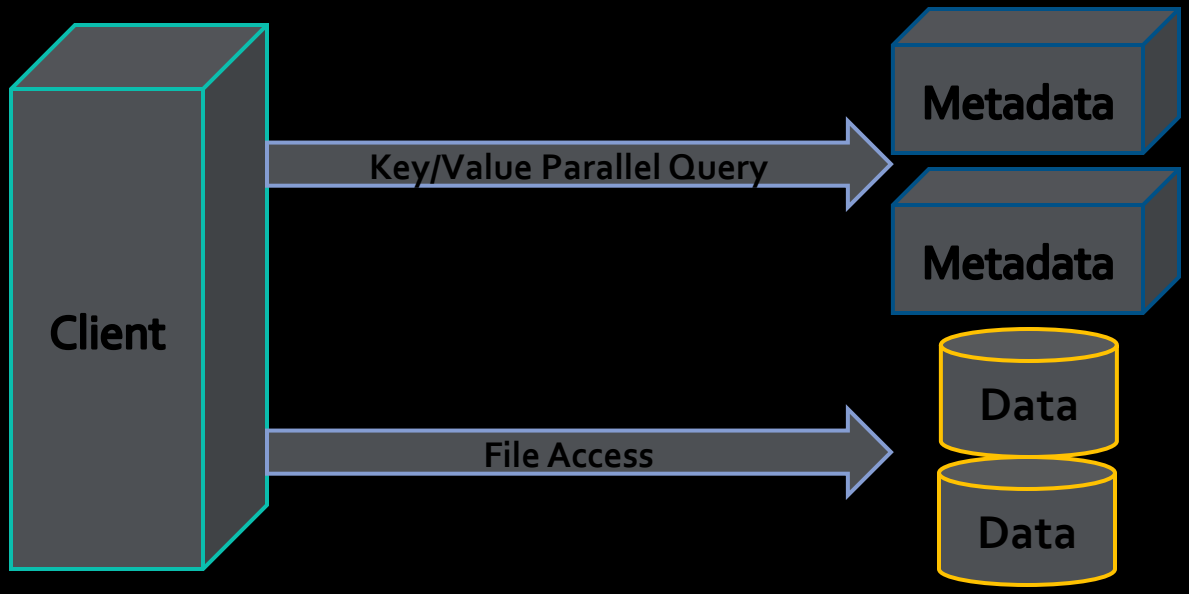
Hierarchical Data Management



Building on Replication

- OrangeFS metadata is extremely flexible
- Moving data
  - Migration (including hardware lifecycle)
  - Archival
  - Data staging
- Locating data
  - Within a directory
  - Across directories
  - Across devices
  - Across systems
  - Across Regions
- Moving computation to data

Attribute Based Metadata Search



- Client tags files with Keys/Values
- Keys/Values indexed on Metadata Servers
- Clients query for files based on Keys/Values
- Returns file handles with options for filename and path

Beyond OrangeFS NEXT

- Extend Capability based security
  - Enables certificate level access
  - Federated access capable
  - Can be integrated with rules based access control
    - Department x in company y can share with Department q in company z
      - rules and roles establish the relationship
      - Each company manages their own control of who is in the company and in department

ParalleX

- ParalleX is a new parallel execution model
- Key components are:
  - Asynchronous Global Address Space (AGAS)
  - Threads
  - Parcels (message driven instead of message passing)
  - Locality
  - Percolation
  - Synchronization primitives
  - High Performance ParalleX (HPX) library implementation written in C++

PXFS

- Parallel I/O for ParalleX based on PVFS
  - Common themes with OrangeFS Next
- Primary objective: unification of ParalleX and storage name spaces.
  - Integration of AGAS and storage metadata subsystems
  - Persistent object model
- Extends ParalleX with a number of IO concepts
  - Replication
  - Metadata
- Extending IO with ParalleX concepts
  - Moving work to data
  - Local synchronization
- Joint effort of LSU, Clemson, and Indiana U.

Community

Learning More

- www.orangefs.org web site
  - Releases
  - Documentation
  - Wiki
- pvfs2-users@beowulf-underground.org
  - Support for users
- pvfs2-developers@beowulf-underground.org
  - Support for developers

Commercial Support

- www.orangefs.com & www.omnibond.com
  - Professional Support & Development team

Questions & Answers