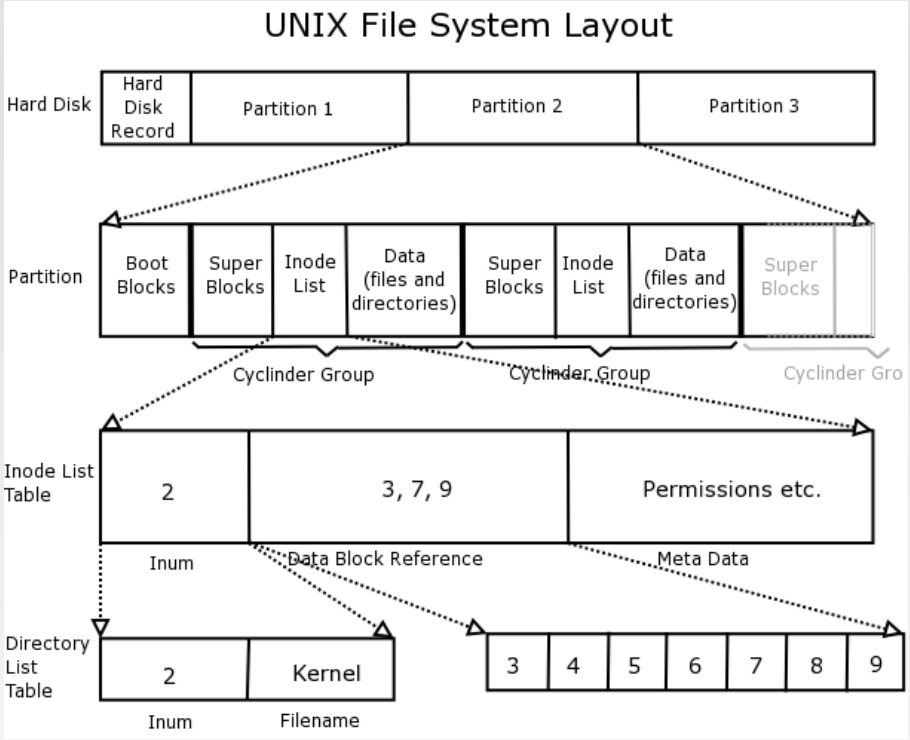
SLASH2

File system for Wide-Area Storage Management



Pittsburgh Supercomputing Center

Paul Nowoczynski, Jared Yanovich, Zhihui Zhang

Need for Wide Area Storage Mgmt

- Geographic replication for valuable data
  (when one site isn't enough!)
- Data generation and analysis often occur at different sites:
  (LHC, LSST, Green Bank, etc.)

Need for Wide Area Storage Mgmt

Cloud Computing

- Maintaining storage environment for applications regardless of run-time locale
- Intelligent staging and shipping of input and output data

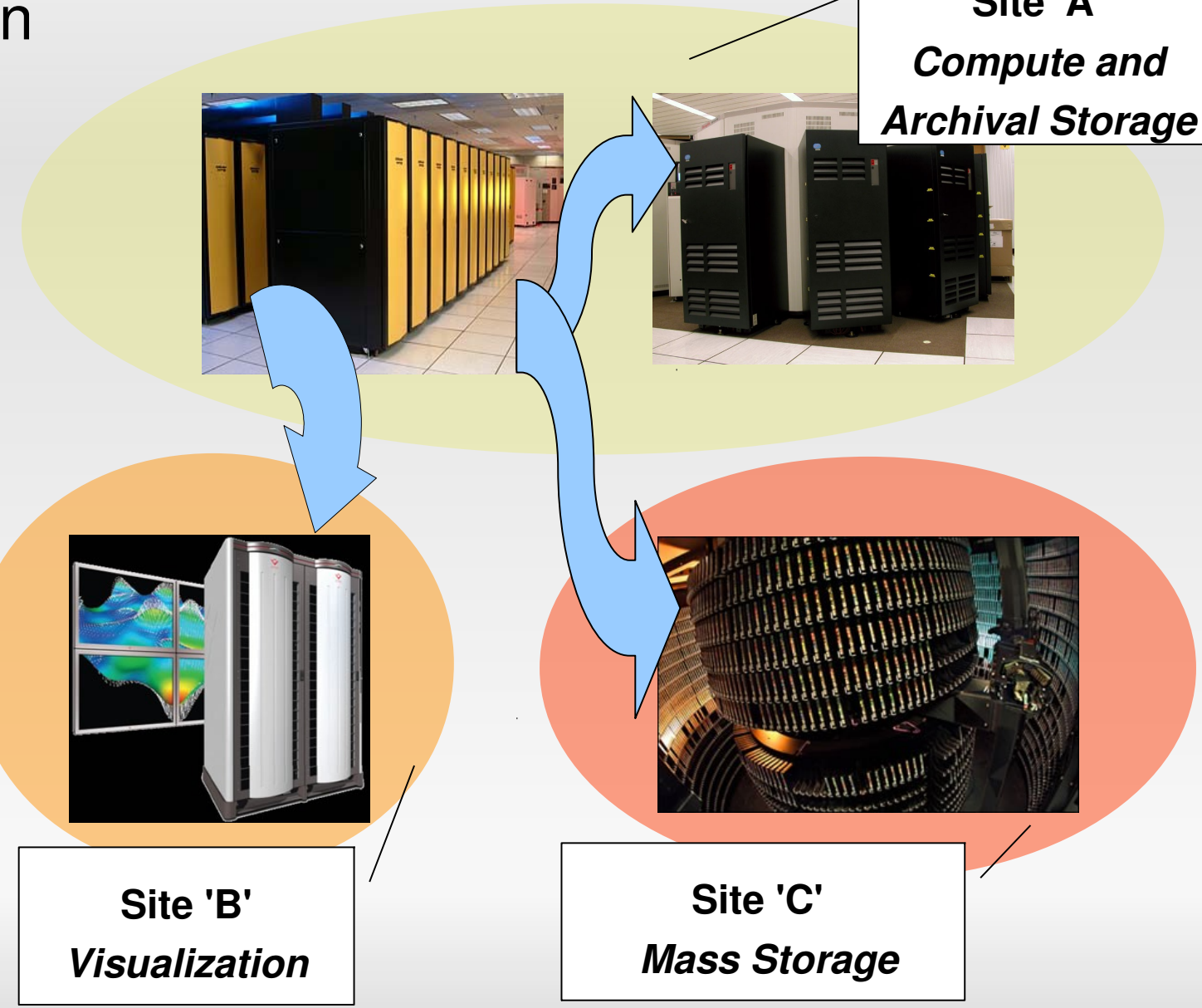
Distributed Research Collaboration

- Presenting a common work environment regardless of location

Wide Area Storage Mgmt within HPC

Data Set Migration

- **Archival storage**
  local or remote
- **Post-processing systems**
  visualization, data analysis
- **A supercomputer with available cycles**



Issues with the current HPC model

- Majority of the effort is forced upon the user:
  - Dealing with environmental inconsistencies
  - Managing data transfers
  - Replica management
- System-managed operations exist in limited capacity
  - iRODS is an exception
- Limited availability of transparent file operations
  - No “global filesystem” which binds storage resources
  - Files must be staged in from remote sites

Objectively speaking..

What do (HPC) users / applications want?

- Universal namespace
- Global data access through standard APIs
  - POSIX
- Local performance (when necessary)
  - Implies tight integration with relevant storage resources
- System-managed data transfers
  - Fully utilize storage and network bandwidth
  - No babysitting!

The reality

- Providing system level uniformity across disparate storage resources is a tall order.
- Many dimensions of heterogeneity exist
  - Varying from the technical to the political

Today's Tools and Methods

Either too much or not enough..

- High-level data mgmt interfaces
  - Data movers (i.e. gridftp, scp)
  - External replica management
- Low-level interfaces
  - Adapting parallel filesystem technologies to the WAN (MC-GPFS, Lustre)

High-level Storage Mgmt Tools

Much of the hard work is placed onto the user:

- Managing data transfers
- Monitoring for failed transfers and issuing retries
- Verification of file integrity
- Adapting to the tools required for data transfer
  - Forced to learn new APIs

High-level Storage Mgmt Tools

Difficult to achieve good performance

- When transferring multi-terabyte data sets, good performance is critical.
- Parallelization or striping across multiple endpoint nodes is necessary, this drastically complicates matters for the typical user.
- Detailed knowledge of the network paths and storage architectures at the source and destination is usually required.

The result is that users are forced to become systems experts OR system experts are needed to aid large data users.

WAN Parallel FS (low-level)

Possibility for system-managed operations exist

- Parallel data migration between sets of OSD's
- Namespace no longer an issue

However, a range of problems exist

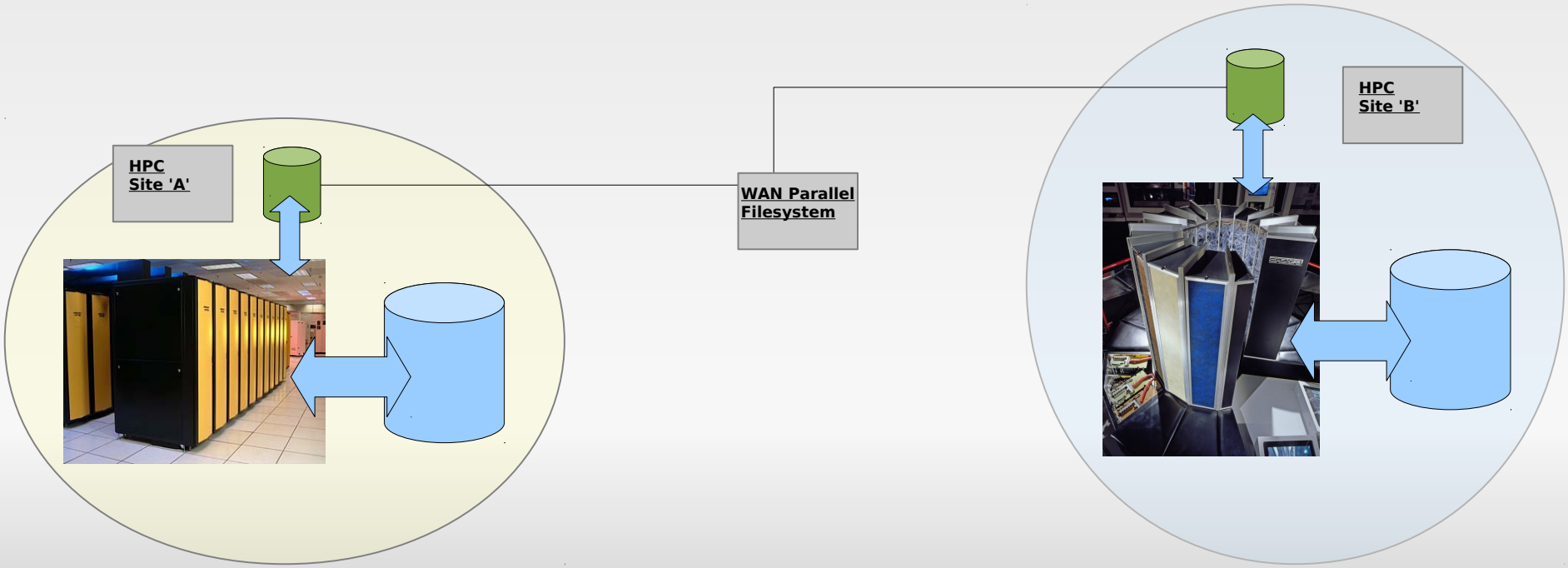
- Vendor lock-in
- Licensing
- Requires strict systems administration procedures between sites
- Increased possibility of cascading outages
- Cannot be integrated with all types of compute or archival resources (portability)

WAN Parallel FS (low-level)

In practice..

Deployed as smaller, auxiliary storage systems to provide a global filesystem for home directories.

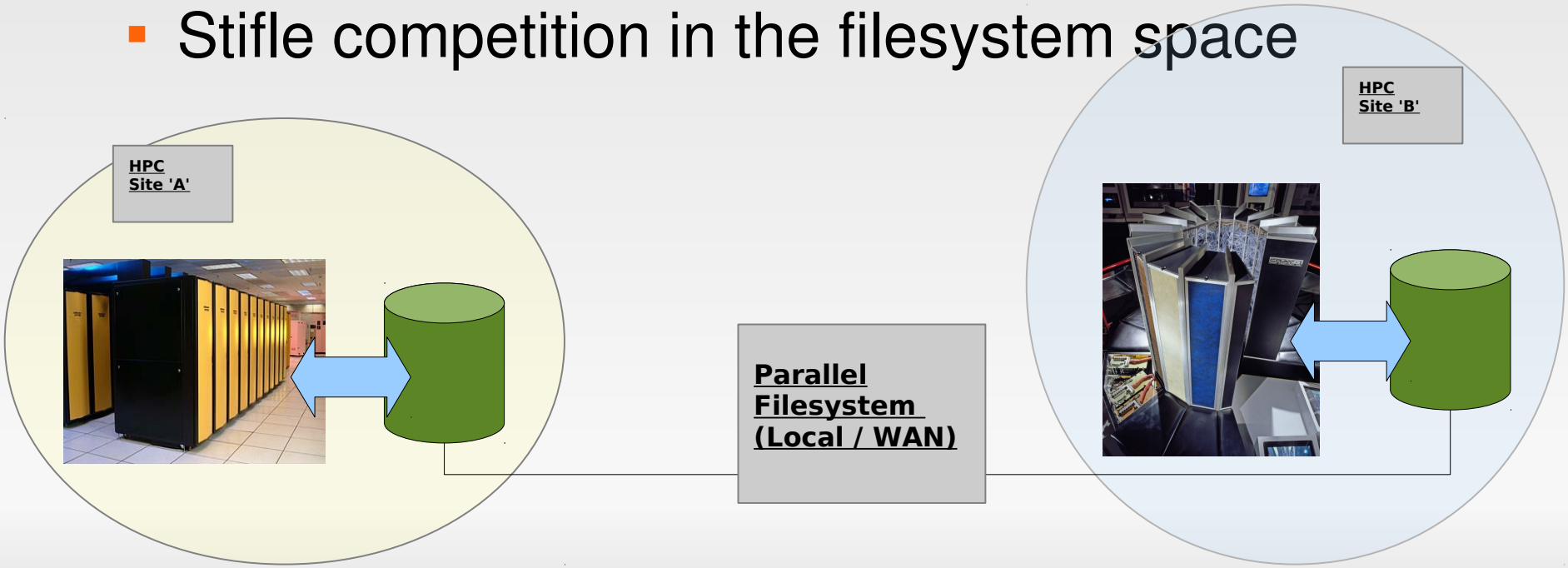
- Does not replace or integrate with the existing HPC storage resources



WAN Parallel FS (low-level)

Can filesystem uniformity be mandated across institutions? Perhaps, but not likely..

- Could negatively impact the procurement process for individual institutions
- Stifle competition in the filesystem space



What is SLASH2?

"Portable filesystem layer designed to enable the inclusion of seemingly disparate storage systems into a common domain to provide system managed storage tasks."

.. or more simply put ...

Data management system designed to aid users who frequently deal with large datasets in grid environments.

Key characteristics

- Highly portable – allow many types of storage systems to be integrated
- POSIX filesystem interface
- Object-based

SLASH2: Background

- Designed implementation by researchers at the Pittsburgh Supercomputing Center
- Inspired by distributed archival caching system developed at PSC in 2004
  - MSST '05 paper on Slash1.
- Funding provided by National Archives and National Science Foundation

How does SLASH2 Contribute to WAN Storage Mgmt?

- Data management activities are performed by the system
  - Monitored by administrators
  - Avoid mistakes made by users
- Provides a common storage protocol which may be ported to a large array of systems
  - Regardless of vendor or storage system class
  - Minimizes dependence on proprietary solutions
  - Aid in integration of new storage and retirement of old
- System stored data checksums
  - Essential for detection of corrupt data
  - May be used for proactive scrubbing of data

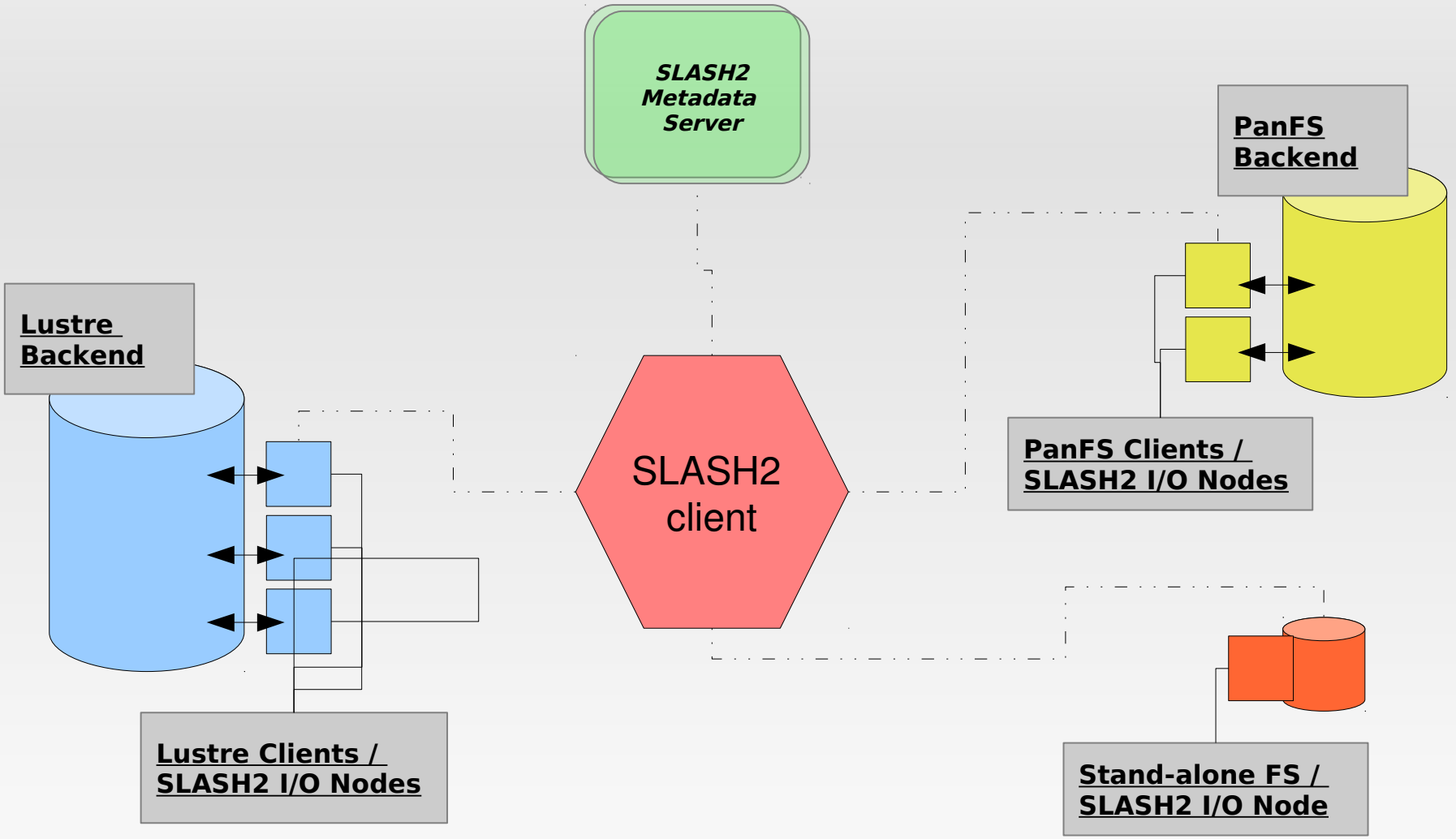
How does SLASH2 Contribute to WAN Storage Mgmt?

- Incorporates essential features into the system

  Move complexity away from the user and into filesystem

  - File replication
  - Integrated replica tracking
  - Inline data verification and rectification
  - Detection of faulty storage endpoints
- Provides a single namespace

SLASH2 Architecture Example



SLASH2 Architecture

- Metadata Server
  - Maintains name → object ID mapping
  - Data residency information
  - Sliver checksums (64bit CRC per 1MB / data)
  - Userspace ZFS under the hood
  - Fully journaled
  - Stateful storage of pending replication requests

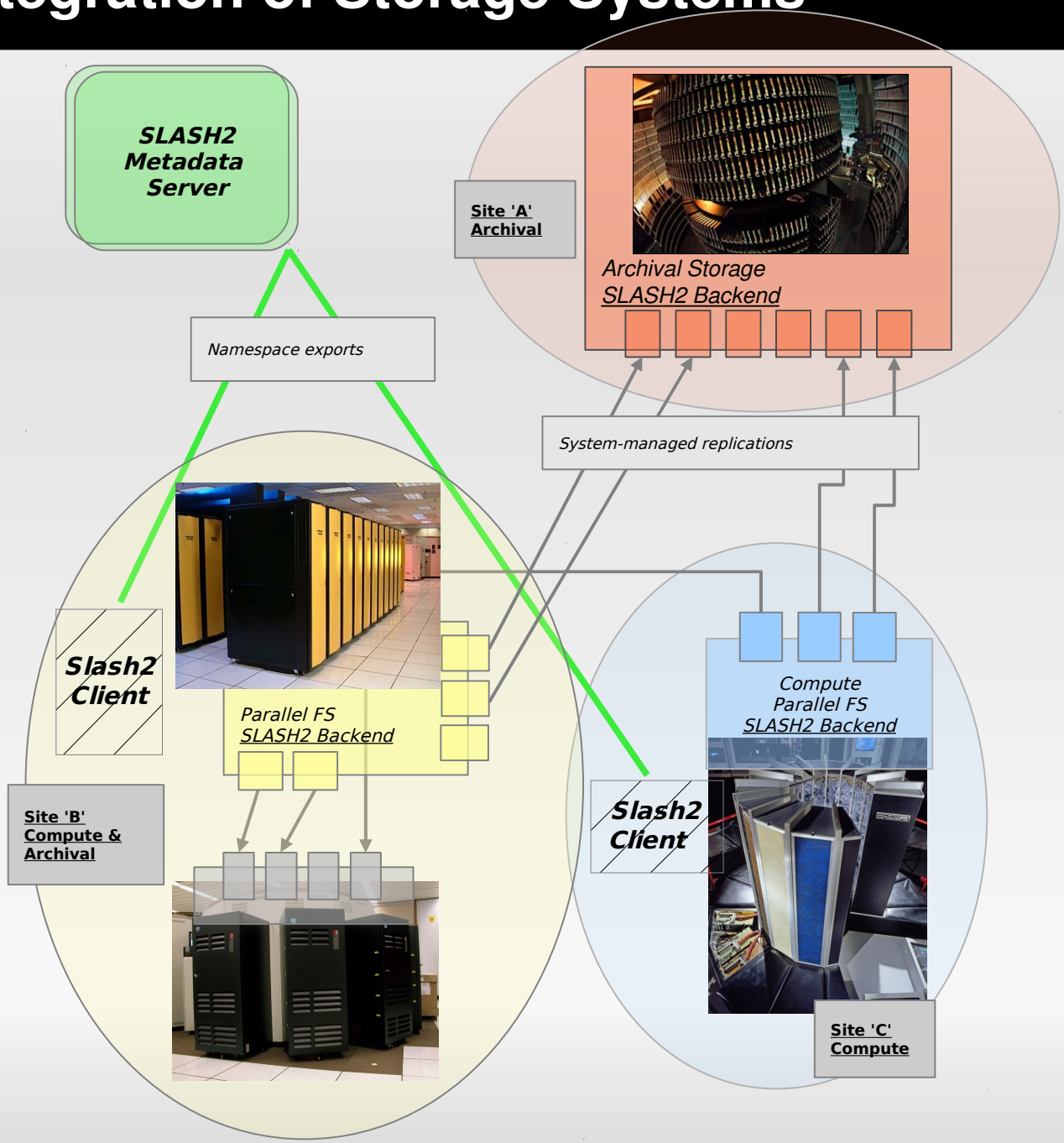
SLASH2 Architecture

- SLASH2 I/O service runs atop an existing filesystem
  - SLASH2 objects stored in local filesystem
  - SLASH2 I/O nodes which mount the same backend filesystem may be used in parallel
    - File I/O
    - Data Replication

SLASH2 Architecture

- SLASH2 client is FUSE-based
- Features of interest
  - Page cache, dnode cache
  - Asynchronous, coalesced writes
  - Readdir+
  - Location aware data retrieval

SLASH2 Architecture:
Logical Integration of Storage Systems



SLASH2: System-Managed File Replication

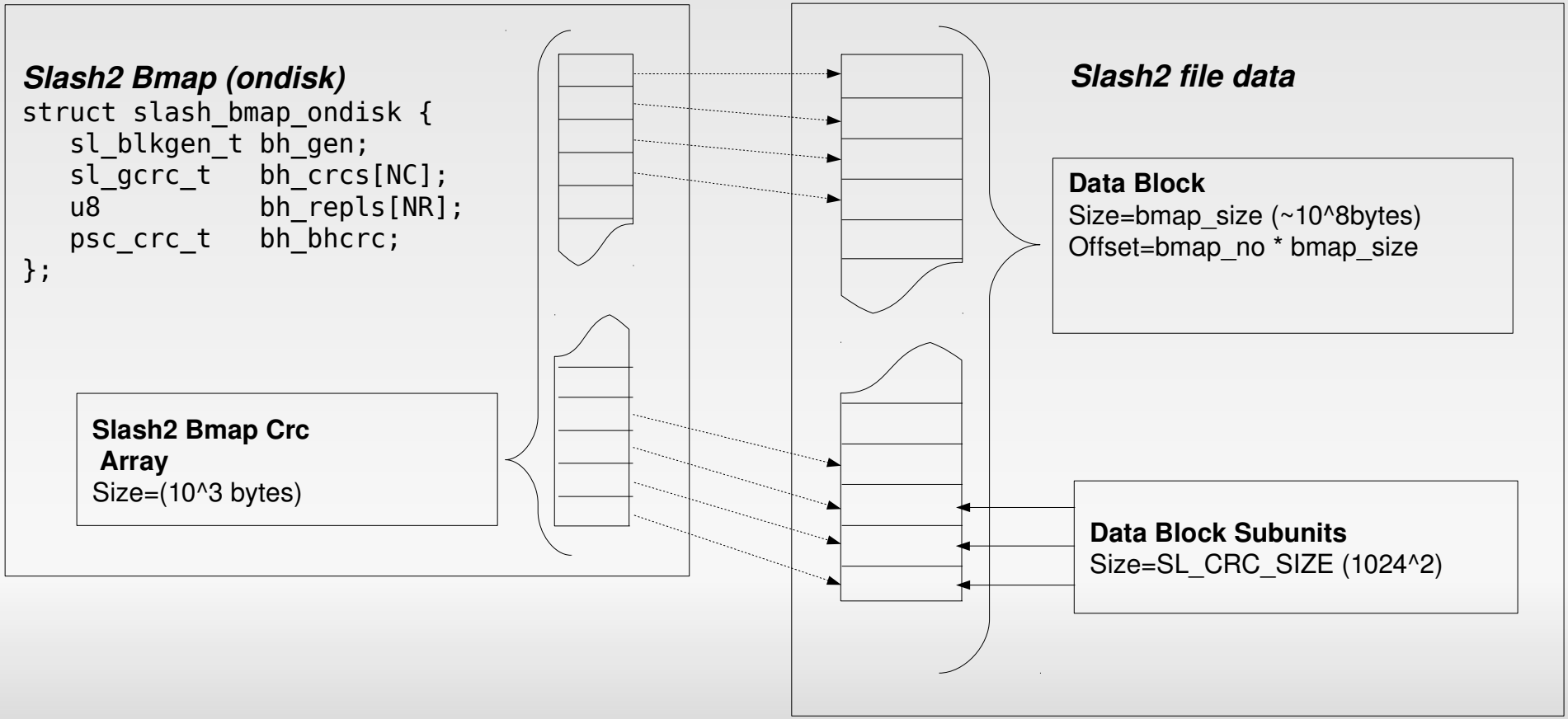
- Ease of use
  - Users specify target files and destination system in a single request - *“copy file set 'D' to Site C's archiver”*
  - If a non-recoverable failure occurs, a system administrator is notified – user intervention is not required.
- Performance
  - When possible, initiates parallel data transfer between data source and destination
  - Uses an effective load-balancing scheme to ensure that a single slow endpoint does not dis-proportionally affect performance.
  - Tuning parameters are held in the system, users are not required to know them

SLASH2: Integrated Replica Management

- Data replicas are systematically maintained
  - Completed replication update a file's residency state
  - Upon overwrite of file data, old replicas are automatically invalidated by the system.
- Replication occurs at block or chunk level, not file level.
  - Partial file overwrites will not invalidate entire file replicas, only the affected block(s).
- Intelligent data retrieval (should data replicas exist)
  - Data retrieval requests are sent to most appropriate storage system
  - Corrupt or missing data may be retrieved from alternate replicas transparently to the application.

SLASH2: BMAP Data Structure

- Central metadata structure, serves to represent a file chunk
- Contains data structures for replica management & data checksums
- Acts as a logical work unit for replication purposes
- Generation numbering for maintaining coherency (invalidating replicas)



Current State of Affairs

- Read() / Write(): stable in most cases
- Replication engine: very close, but bugs persist
- Namespace operations have been thoroughly tested
  - Coherency model similar to NFS – no leased locks
- Security
  - UID / GID based
  - MDS has NFSv4 ACL support through ZFS
  - RPC validity is determined through a shared-key mechanism

Development for 2010 - 2011

- Test and evaluation of a distributed metadata system prototype using *Eventual Consistency*
  - Aimed at providing reliable but asynchronous namespace mirroring between metadata servers.
  - Implements a new algorithm where the metadata servers may modify a single namespace simultaneously without creating conflicts.
    - Modification logs are passed amongst the metadata servers and applied in a deterministic fashion
    - Ensures that metadata servers are always 'approaching synchronization, though they may not be in sync at any given moment.

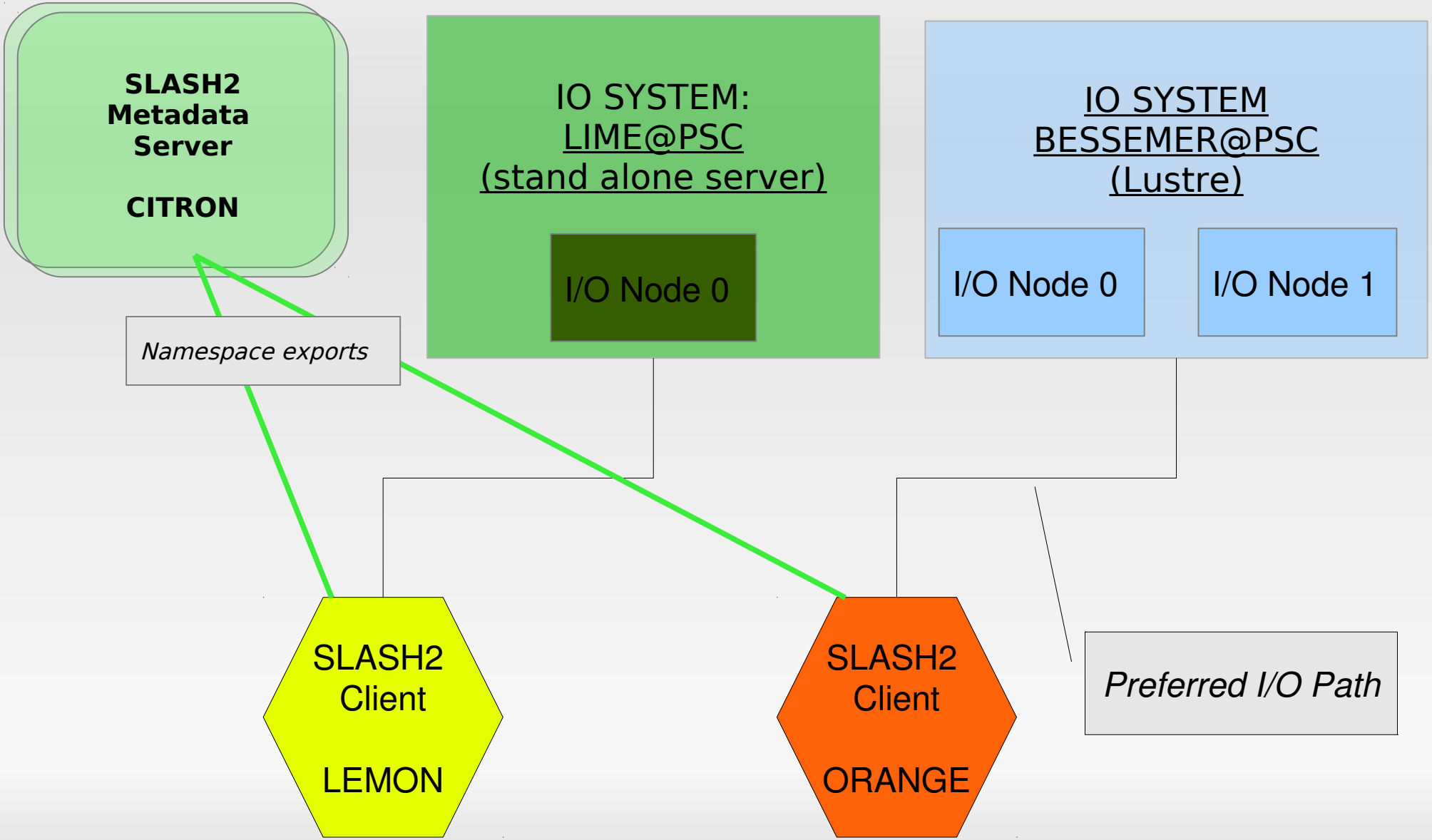
Development for 2010 - 2011

- Secure Authentication
  - Design and approach are still being discussed
  - Node and user authentication
  - Kerberos?
  - Borrow concepts from the IU / Data Capacitor model?
  - *What are the opinions of the TeraGrid community?*
- More Debugging!!
  - Deployment of automated test system (tsuite)

Demo Configuration

- Single Metadata server
- 2 Clients
- 2 I/O Systems (bessemer@PSC, lime@PSC)
  - Bessemer@PSC - 2 I/O nodes

Demo Configuration



Demo Configuration

- Only one site at the moment (@PSC)
- Would love to add more!

```
site @PSC {
        site_desc    = "Pittsburgh Supercomputing
Center";
        site_id              = 2;
        resource bessemer {
                desc  = "DDN9550 Lustre Parallel Fs";
                type   = parallel_fs;
                id       = 0;
                ifs      = 128.182.99.124,
                           128.182.99.123;
                fsroot= /bessemer/pauln/s2io/;
        }
        resource lime {
                 desc    = "Stand-alone I/O server";
                 type    = standalone_fs;
                 id      = 1;
                 ifs     = 128.182.99.27;
                 fsroot = /s2io;
        }
        resource citron {
                desc  = "WVU Testbed MDS";
                type   = mds;
                id       = 99;
                ifs      = 128.182.99.29;
                jrnldev = /dev/md1;
        }
}
```

Demo Items

- Replication test
- Create test
- Stat test
- Build test

Slash2 Website

- http://quipu.psc.edu/slash2
- Under heavy construction.. Will have development resources soon!
  - Source code download
  - Access to svn repository
  - Bug tracking and submission
  - Mailing list