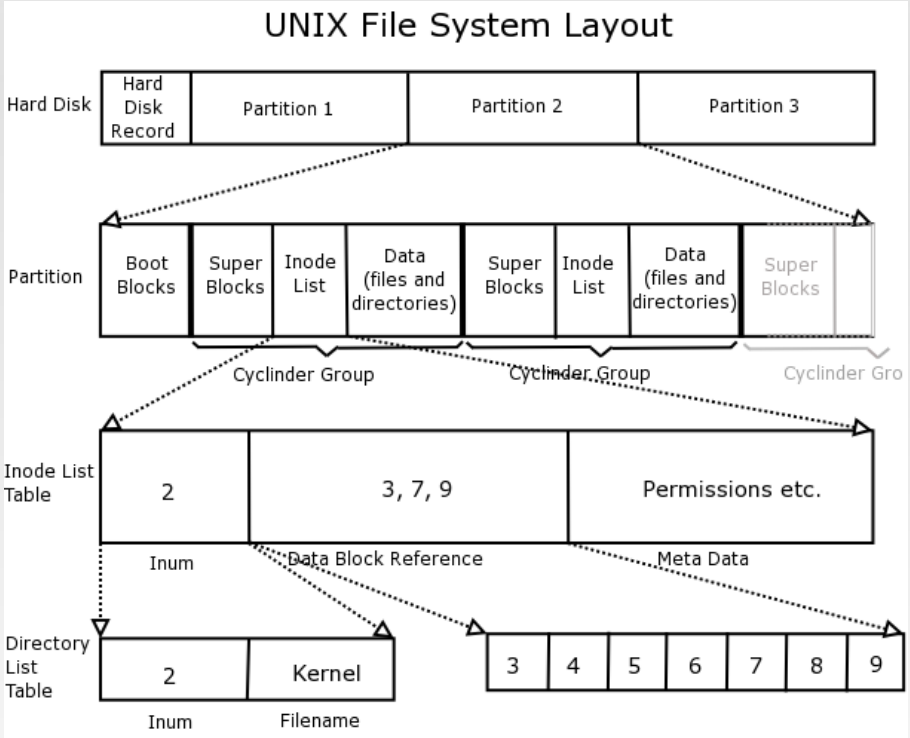
SLASH2

Scalable Lightweight Archival Storage Hierarchy 2



Pittsburgh Supercomputing Center in Collaboration with West Virginia University

SLASH2: Introduction

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 of SLASH2

- Highly portable – allow many types of storage systems to be integrated
- "Look and feel" similar to a Unix filesystem

SLASH2: Background

- Designed implementation by researchers at the Pittsburgh Supercomputing Center
- Inspired by distributed archival caching system developed at PSC in 2004
- Work has been ongoing since fall of 2006.
- Funding provided by National Archives and National Science Foundation

PSC / WVU Collaboration

- Collaboration established between PSC and WVU in fall of 2008
- Implemented the first SLASH2 wide-area testbed between Pittsburgh and Morgantown
  - Demonstrate features and capabilities over wide-area network
  - Inter-agency cooperation in establishing common filesystem
- WVU researchers are contributing to the ongoing testing and evaluation effort

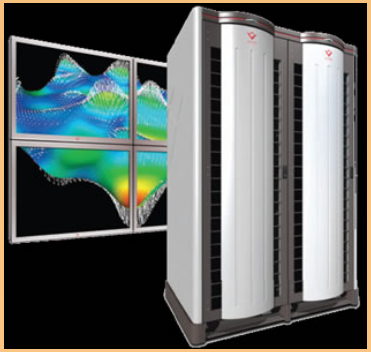
What Problems does SLASH2 Address?

SLASH2 targets the complexity encountered by large data users who commonly operate in heterogeneous storage system environments.

- Scientific users dealing with large data generally operate in these environments..

Typical Heterogeneous Storage Environment

Site 'A'
Compute and Archival Storage



Site 'B'
Visualization

Site 'C'
Mass Storage

Why is Storage Heterogeneity Common?

- Storage systems have properties suited to their role.
- Users generally straddle at least 2 storage system types to accomplish their tasks.
- Data grids are heterogeneous by default..

- **Archival storage**
  - Speed: Slow, usually tape based
  - Extremely high capacity
- **Parallel filesystem**
  - Speed: Very fast, used for checkpointing application state on supercomputers
  - Moderate capacity
  - Data is archived or deleted
- **Storage for sensor-based systems** (satellite images, Large Hadron Collider, LSST)
  - Speed: Moderate to fast
  - Capacity is limited, data must be offloaded to archival storage.
  - Data is rarely deleted

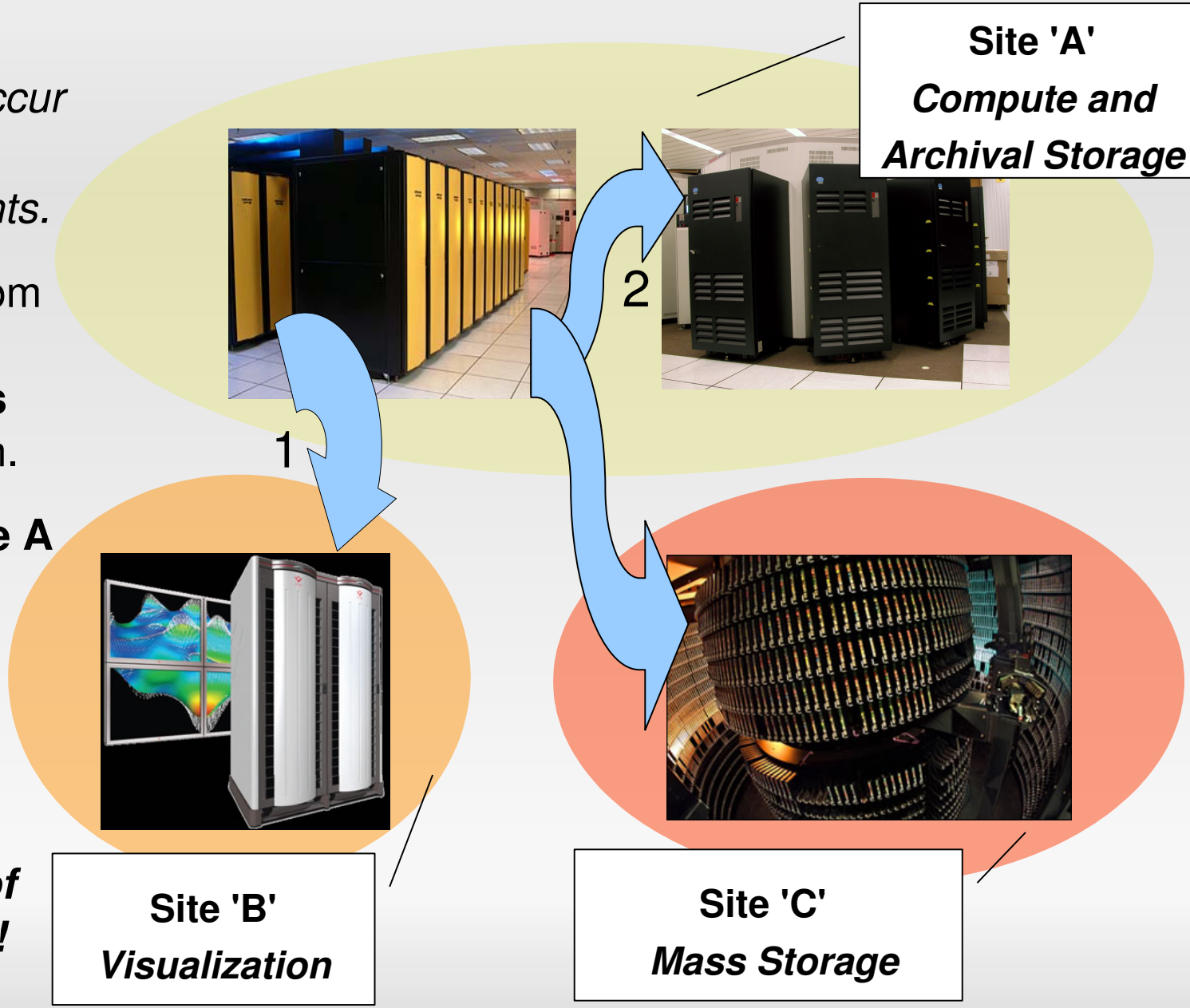
Example of a Common Storage Operation

Here are two typical operations which occur in heterogeneous storage environments.

1) Copy data set 'D' from **Site A's** compute resource to **Site B's** visualization system.

2) Archive 'D' from **Site A Compute** to two geographically separated archival systems.

User's think of file transfer in terms of logical operations!



What Issues do Large Data Users Face?

1. Lack of convenience - 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
   - Manual tracking of file replicas

What Issues do Large Data Users Face?

2. Difficulty achieving 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.

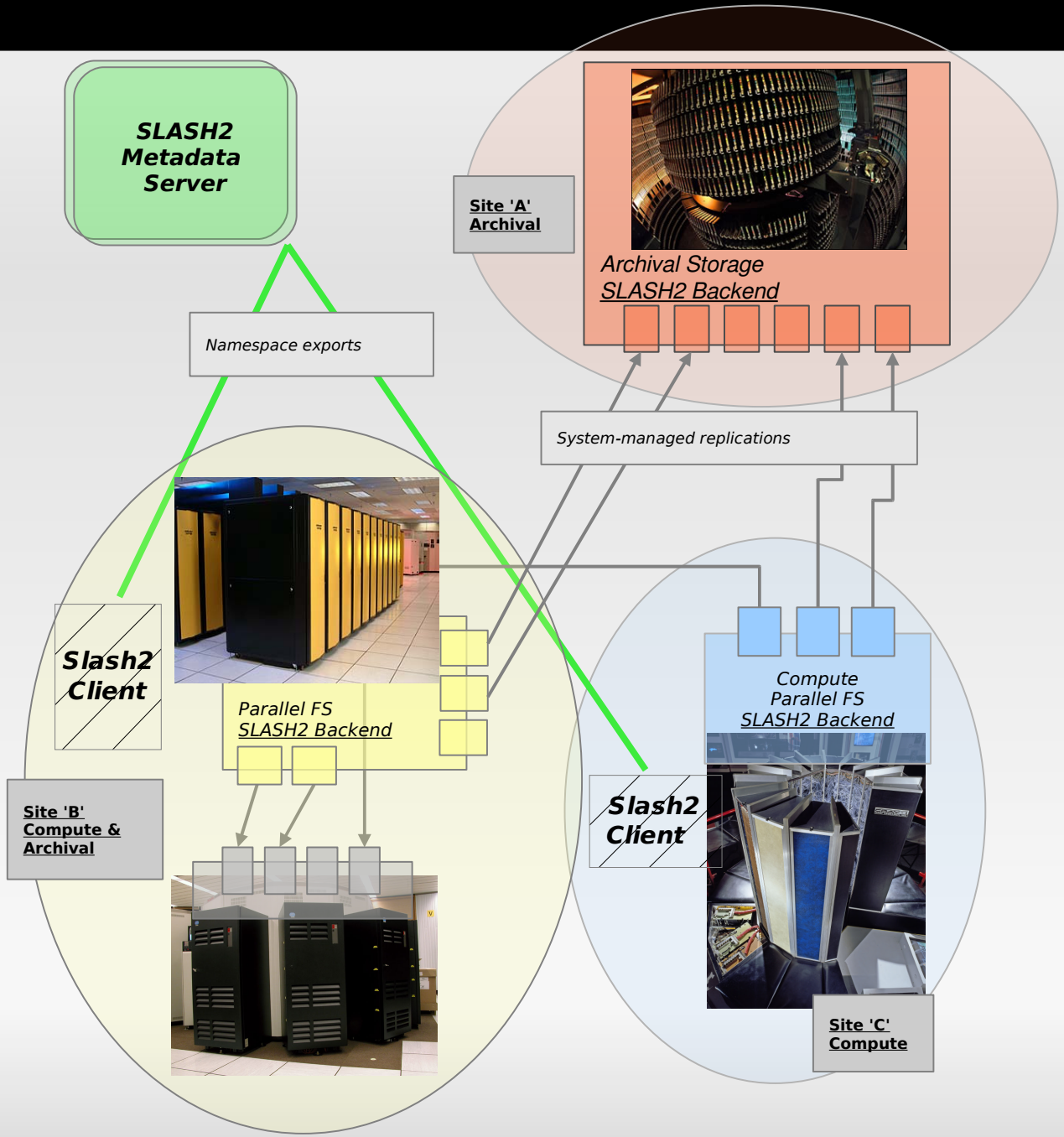
The result is that users are forced to become systems experts OR system experts are needed to aid large data users.

How does SLASH2 Address these Issues?

- Incorporates essential features into the system
  - Move complexity from the user and into filesystem
- What system managed features are available?
  - File replication
  - Integrated replica tracking
  - Inline data verification and rectification
  - Detection of faulty storage endpoints

Logical Integration of Storage Systems

- External metadata server provides centralized management
  - Exports common namespace to applications
  - Central store for data checksums
  - Coordinates data replication activities



SLASH2: System-Managed File Replication

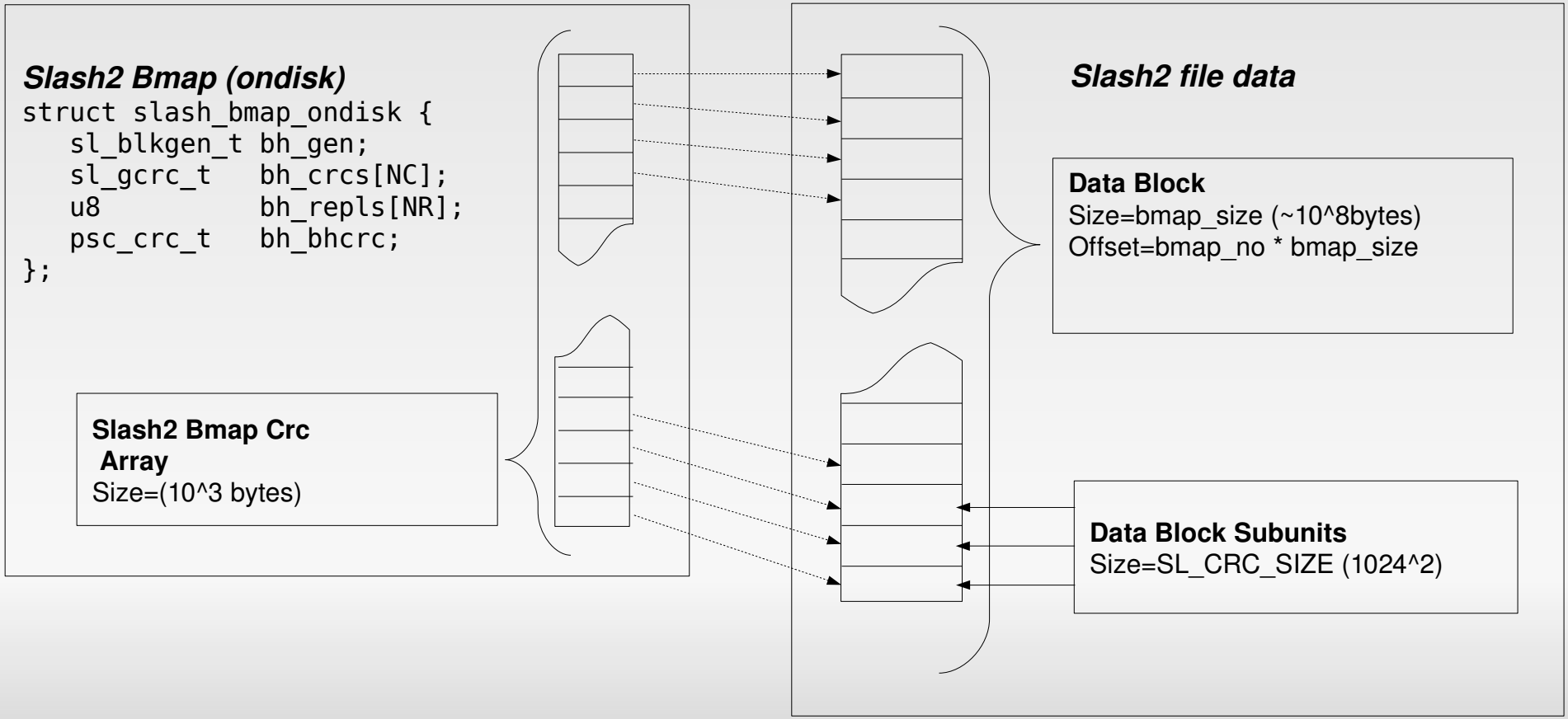
- Ease of use
  - Users specify target files and destination system in a single request - *“copy file set 'D' to Site C's archiver”*
  - If a non-recoverable failure occurs, a system administrator is notified – user intervention is not required.
- Performance
  - When possible, initiates parallel data 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 of 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)



SLASH2: Centralized Namespace

- Stand-alone metadata server provides common namespace to applications.
- Namespace is mountable by the SLASH2 filesystem client based on the FUSE module (Filesystem in Userspace).
- FUSE mountpoint provides POSIX file access for most operations.
- Operations, such as those for file replication, require stand-alone utilities – no POSIX system interface exists for them.

Storage of High-Value Data: How does SLASH2 Contribute?

- Data management activities are performed by the system
  - Monitored by administrators
  - Avoid mistakes made by users
- System stored data checksums
  - Essential for detection of corrupt data
  - May be used for proactive scrubbing of data
- 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

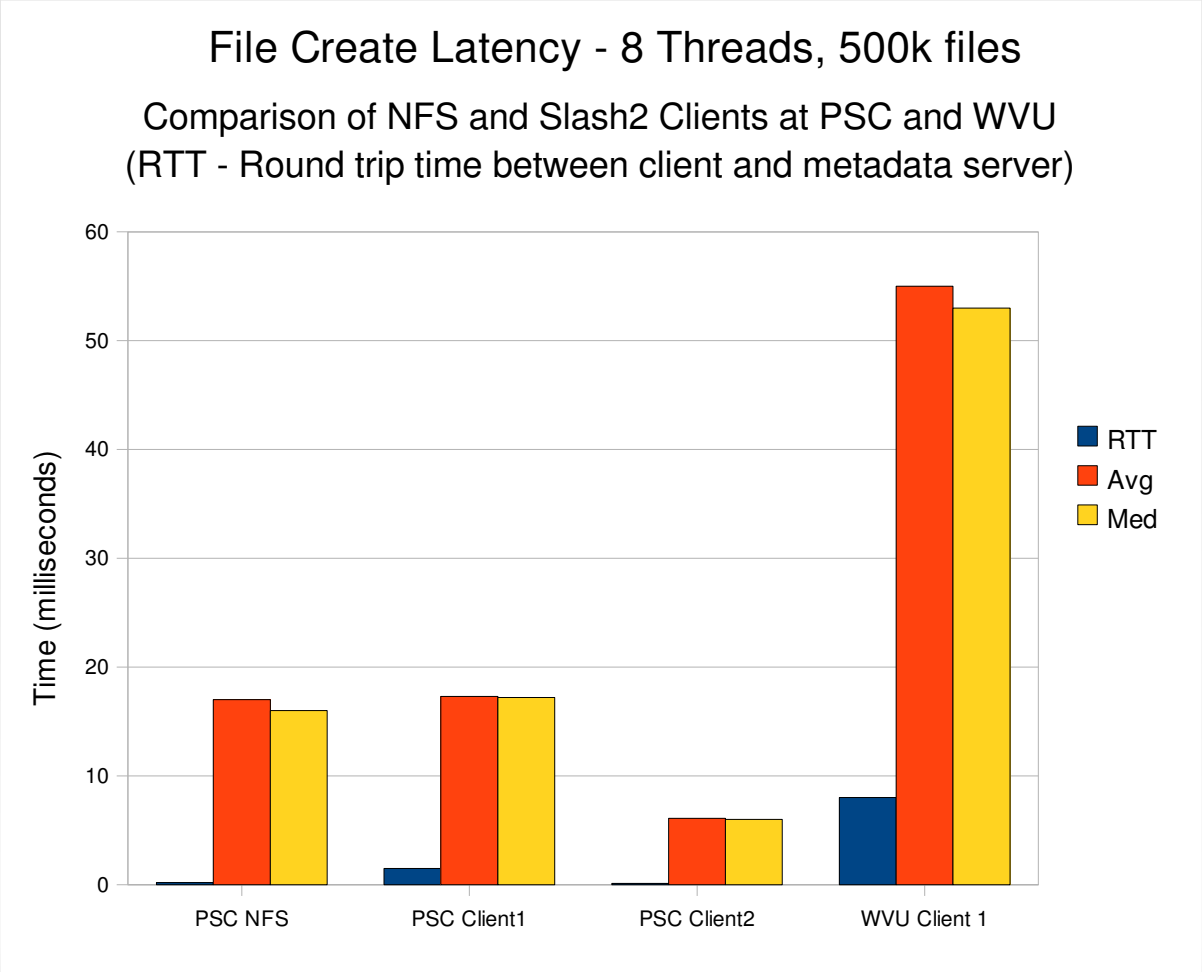
Fruits of the PSC / WVU Testbed

- The first half of the work period was spent verifying correctness of the namespace
  - Namespace operations between PSC and WVU exhibit behavior as expected of a distributed filesystems.
  - Coherency model is similar to that of NFS – using timeout on cached objects as opposed to leased locks.
  - Through systematic test methods, millions of files and directories have been created, deleted, and verified.

Fruits of the PSC / WVU Testbed

File create performance results

- Blue bar (RTT) represents the round-trip time between the client and server.
- For similar RTT (PSC client 1, PSC NFS), SLASH2 outperforms NFS by a factor of 3.
- An 1.38 ms increase in RTT causes operation latency to increase by a factor of 3.
- Higher RTT's hurt performance, as in the case with WVU Client. However, 50 ms is quite usable for most interactive operations.



PSC / WVU Testbed: Next Steps

- Between now and the Oct. '09 – Lots to do!
  - Rigorous testing of SLASH2 read and write capability through Linux / FUSE.
    - Includes verification of system managed data checksums
    - Deal with local and remote data
    - Verify the functionality of SLASH2 data corruption detection and transparent rectification.
  - Testing and performance testing of file replication capabilities.

PSC / WVU Testbed: Possible Future Research

- Test and evaluation of a distributed metadata system prototype designed PSC
  - 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.

Would drastically improve performance by lowering clients' RTT to the metadata system.