Applicability of object-based storage devices in parallel file systems

Pete Wyckoff

Ohio Supercomputer Center

pw@osc.edu

HECIWG 21 aug 06

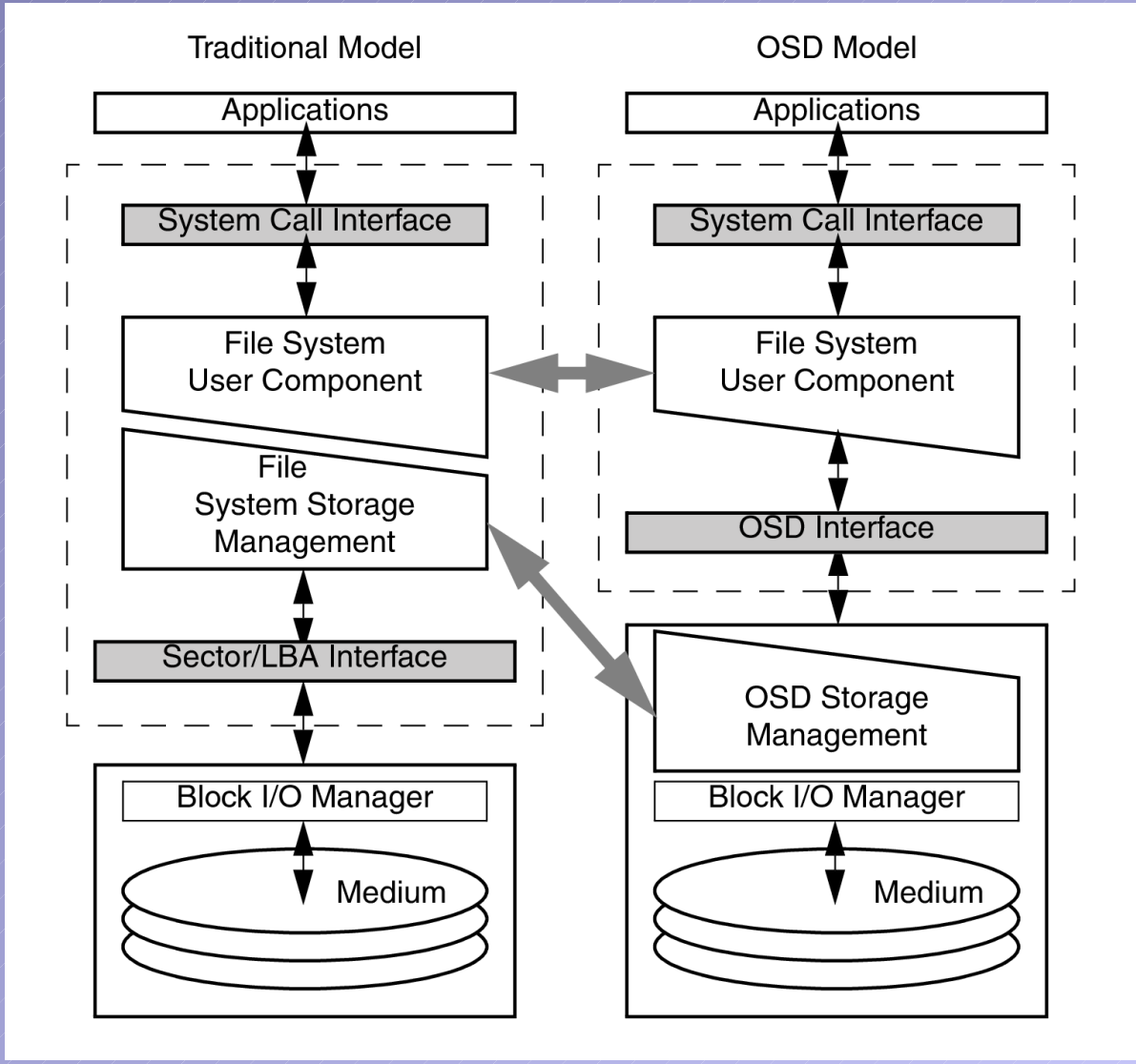
Vision

- Processors faster, disk densities up, but IO rates flat
- Leverage intelligent peripherals to improve performance
- OSDs offer higher-level semantic interface
- Secure, direct access of storage by clients
- Our work
  - Examine role of OSDs in parallel file systems
  - Analyze trade-offs of using OSDs for various aspects of parallel FSes
  - Develop extensions required for efficient use of OSDs in HPC parallel environments

OSD Background

- T10 specification
- Prototypes
- Simulators

- Pure target
- Security model



Mapping Data to Objects

- Block-based "objects" are 512 bytes
- OSD objects could be files
- For striped files, object is stripe, or stripe set?
- Collection feature allows grouping objects
  - flush, remove, list, access control
- Delegating object creation to clients
  - maybe using collection container
- When to create objects on create?
  - lazy, preallocate

Metadata

- OSDs store *attributes* with objects
- How do parallel FS attributes map to these?
  - uid, gid, perm, [acm]time, type
    - mtime of object vs mtime of file
  - size, link target, dfile count, dirent count, dir hint
  - xattrs
  - metafile contents (datafile handles, distribution)
  - directories themselves?
- Managing collective behavior
  - object allocation, fsck, rebalancing
  - use OSDs for metadata management too?
- Select objects by attribute
  - data-introspective extensions

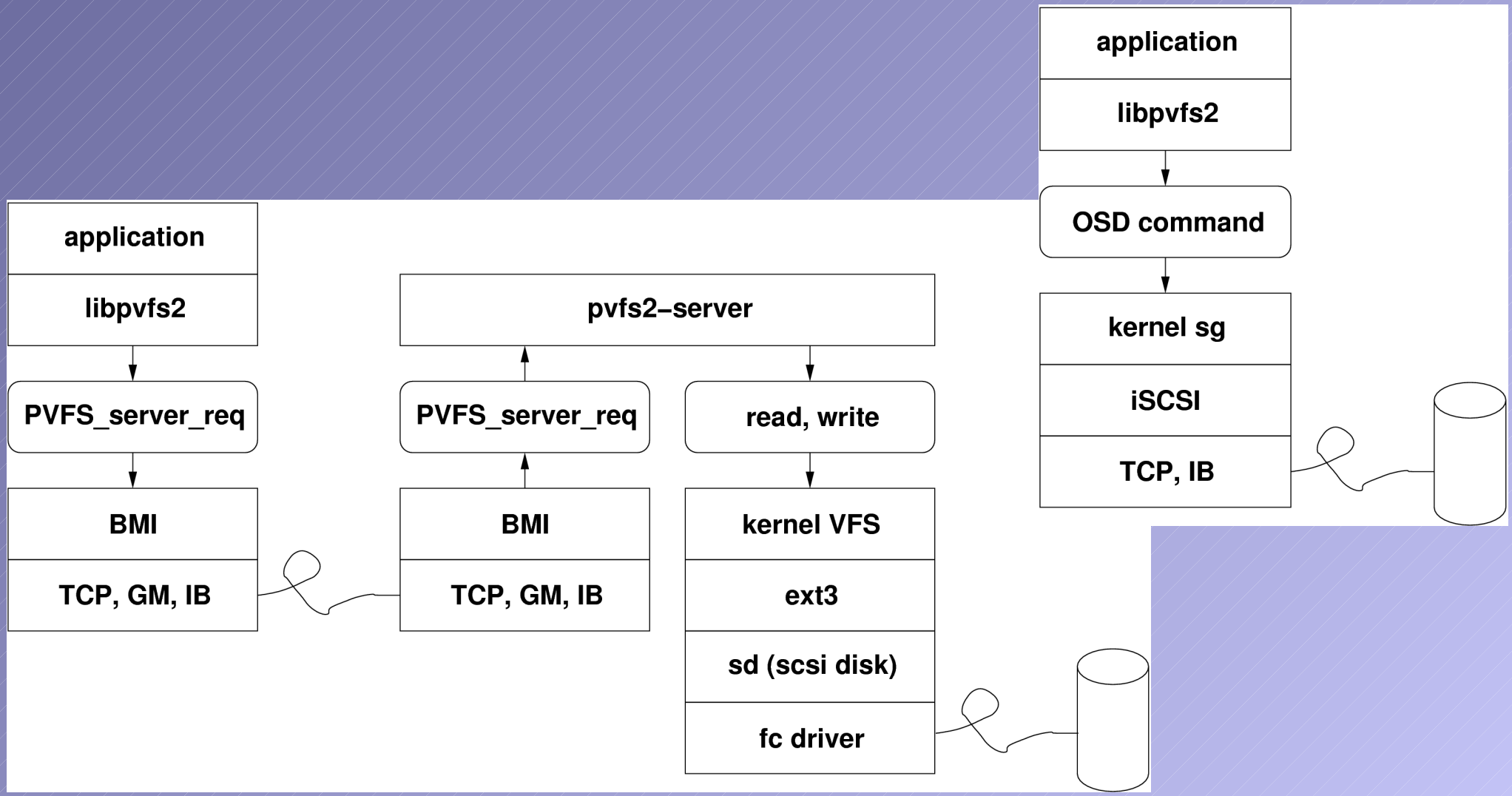
Other Topics

- Transport
  - choice of RDMA direction based on usage intent
  - link flow control and object information
  - drive prefetch, write-behind by object info
- Caching
  - attributes can be used to build DLM
  - versioning FS
  - dial-a-consistency approach
- Reliability
  - manage disk cache explicitly
  - auto-checksum or auto-duplicate some objects

Implementation

- All in context of PVFS
- Open source, LGPL license, release often
- Maximize benefit to other researchers and production implementations
- Particular components
  - specific client-side transport protocol for OSDs
  - separate out "meta" and "data" operations
  - metaserver may need to initiate "data" operations
- OSD implementations
  - iSCSI initiatiors (IBM user, stock kernel)
  - iSCSI target (IET kernel), stgt (user/kernel)
  - OSD initiator (IBM, Intel)
  - OSD target (Intel, IBM (no source))

Protocol Stacks



Conclusion

- Enable higher semantic interface to storage
- Avoid middle-box interference on data path
- Understand roles of clients, metadata servers, IO servers, lock servers, etc in parallel FSes

- Cooperation welcomed