Structuring PLFS for Extensibility

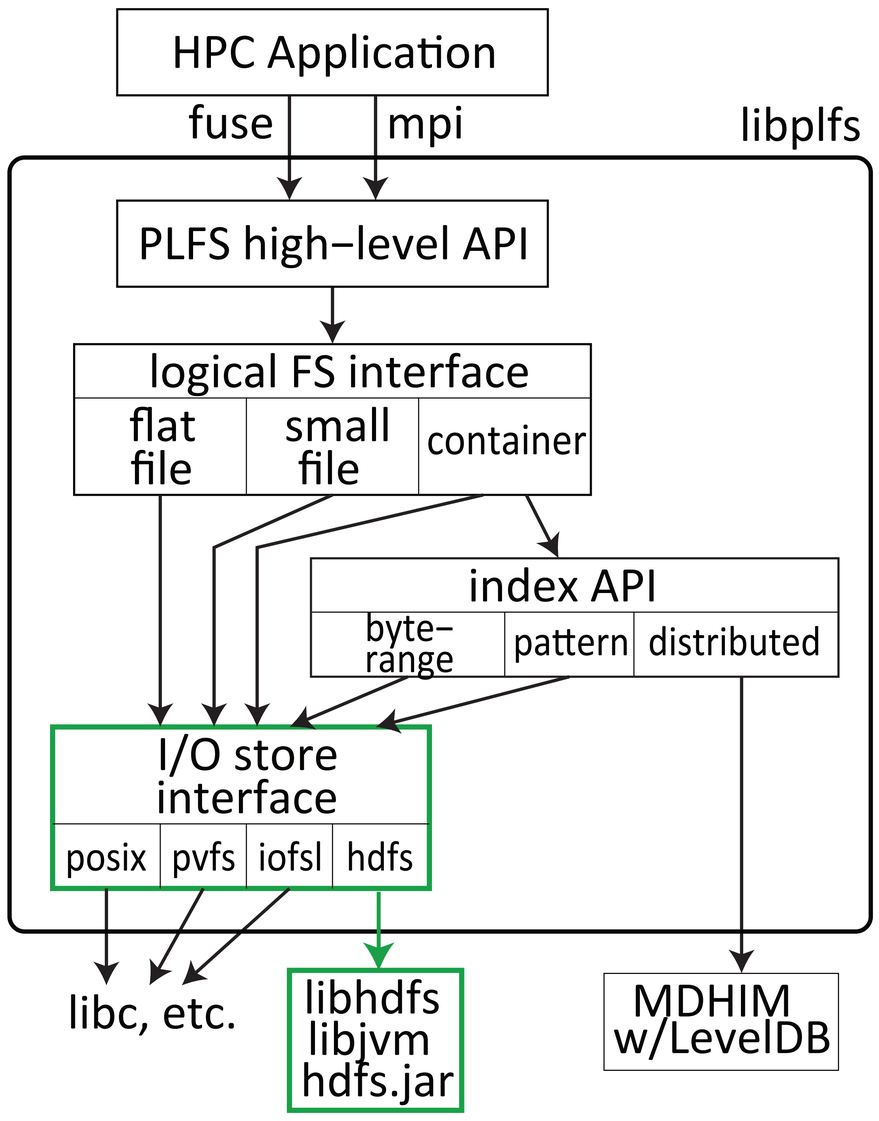
Chuck Cranor, Milo Polte, Garth Gibson (Carnegie Mellon University)

Problem

- HPC applications checkpoint to a single shared file
- The filesystem must:
  - › Make new checkpoint files visible on all nodes at creation time
  - › Support highly concurrent writes at widely varying offsets
- Cloud storage systems such as the Hadoop File Systems (HDFS) are optimized for cloud-based applications such as Map Reduce
- POSIX I/O semantics are relaxed to improve performance:
  - › Only one node can have a file open for writing
  - › All writes are append-only; re-open = truncate
- Storage allocated to HDFS does not work for N-1 checkpointing
  - › Extensibility in PLFS can help solve this problem

Extensibility in PLFS

- Logical FileSystem: supports multiple filesystem types
- Container Index API interface: easily change indexing schemes
- I/O Store: uses non-POSIX backends

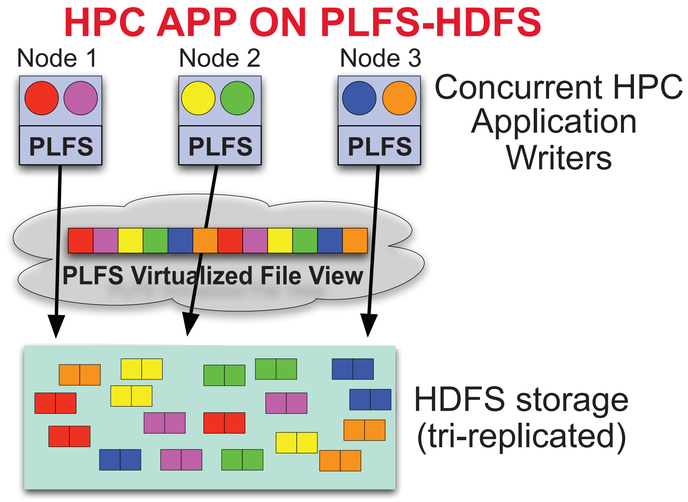


HDFS I/O Store

- Must map PLFS I/O Store calls to HDFS API, 3 cases:
  1. Direct mapping: read maps to hdfsPread()
  2. Mapping with minor adjustments
     - POSIX file descriptor to hdfsFile handle structure
     - owner/group int ids vs. owner/group strings
     - POSIX file/dir creation API sets permissions too, HDFS does not
  3. Not possible (device files, symbolic links)

Parallel Log Structured Filesystem (PLFS)

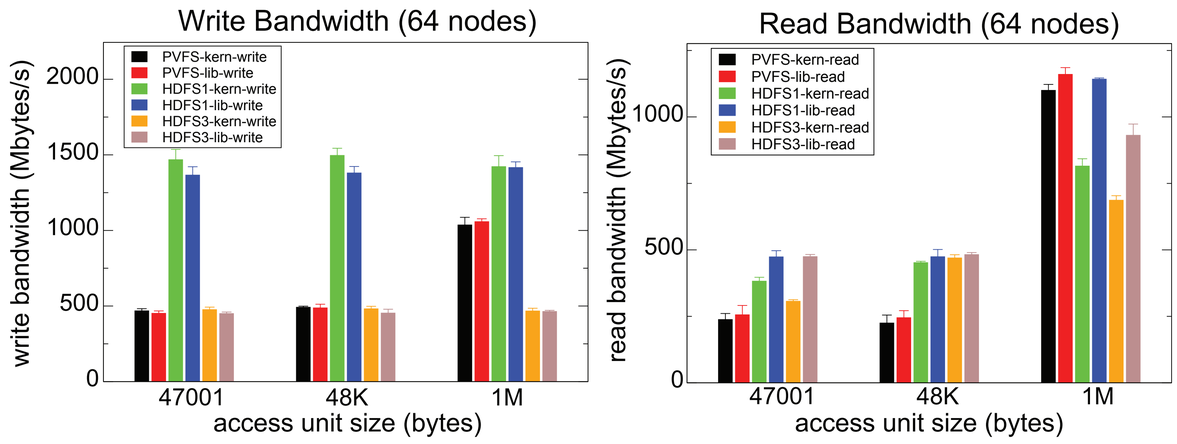
- FUSE or MPI-based filesystem that decouples concurrent writes by logging each node's writes separately
  - › Improves performance by avoiding sharing bottlenecks
  - › PLFS's log structured writes fit the limited semantics of HDFS cloud storage



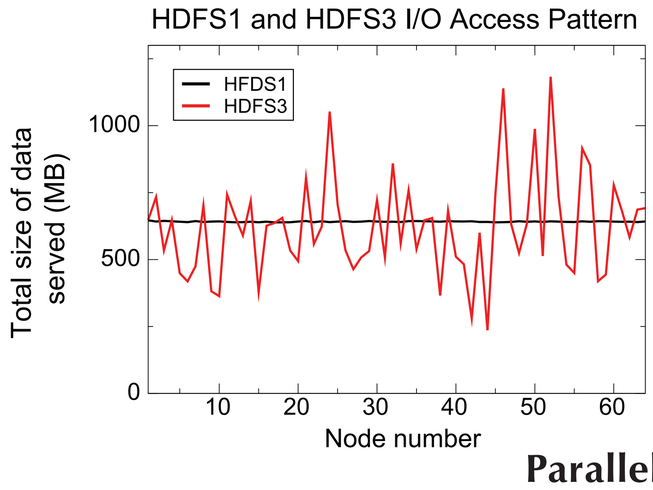
Results

Platform: Marmot PRObE cluster

- 1.6GHz AMD Opteron dual processor, 16GB memory, 1GE
- Hadoop HDFS 0.21.0, FUSE 2.8, PLFS, OrangeFS 2.8.4 (PVFS)
- LANL test_fs N-1 benchmark with 47001, 48K, or 1M ranges
- 6 test cases: PVFS, HDFS1 (no replication), HDFS3 (3 way replication) through a kernel mount point and a library API



- PLFS/HDFS is roughly comparable to PVFS
  - › writes: HDFS1 always writes to local disk (fast, no network)
    - HDFS3 has 3x replication overhead
    - PVFS network limited with small access size
  - › reads: HDFS benefits from PLFS' log structured writes
    - Kernel buffer cache hurts 47001 reads due to page alignment
    - 1M HDFS suffers from extra overhead of Java/data copies
    - 1M HDFS1 outperforms HDFS3 due to balanced I/O pattern



Carnegie Mellon University

Parallel Data Laboratory