PLFS/HDFS: HPC APPLICATIONS ON CLOUD STORAGE

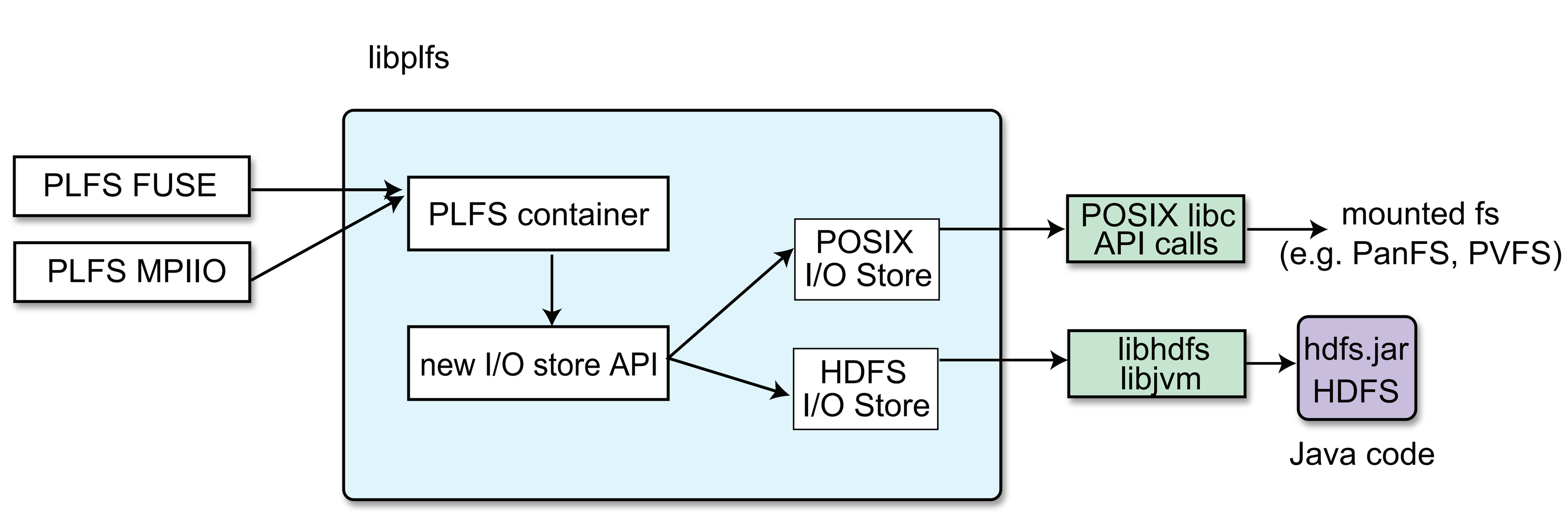
Chuck Cranor, Milo Polte, Garth Gibson (Carnegie Mellon University)

PROBLEM

- Parallel High Performance Computing (HPC) applications checkpoint progress to a single shared file on a networked filesystem
- The filesystem must:
  - Make newly created checkpoint files visible on all nodes at file creation time
  - Allow nodes to have concurrent write access at varying offsets to the checkpoint file
- 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 at a time
  - All writes are append-only
- Storage resources allocated to HDFS cloud storage cannot be used by HPC applications for N-1 checkpointing

PLFS I/O STORE ARCHITECTURE

- Insert new I/O store layer into PLFS to allow multiple types of backing filesystems to be used (including non-POSIX ones)
- Current list of I/O stores: POSIX, HDFS, PVFS, IOFSL (in progress)
- HDFS I/O store module uses libhdfs API for log I/O
  - Hadoop's libhdfs uses Java Native Interface (JNI) to provide C/C++ access to HDFS Java methods
  - Links a Java Virtual Machine into PLFS

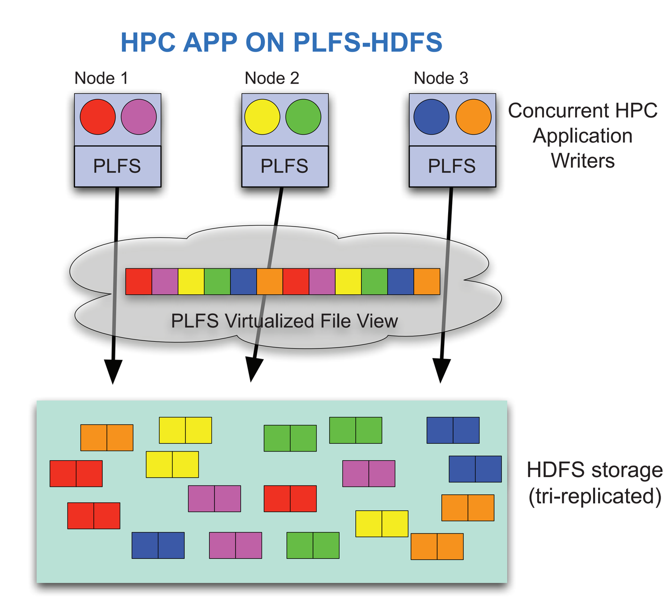


HDFS API AND PLATFORM ISSUES

- 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)
- HDFS Java platform has issues when used by threaded/forking C++ applications such as PLFS and is difficult to debug due to multiple domain crossings
  - e.g. write: application → kernel → FUSE daemon → JVM → HDFS daemon

PARALLEL LOG STRUCTURED FILESYSTEM (PLFS)

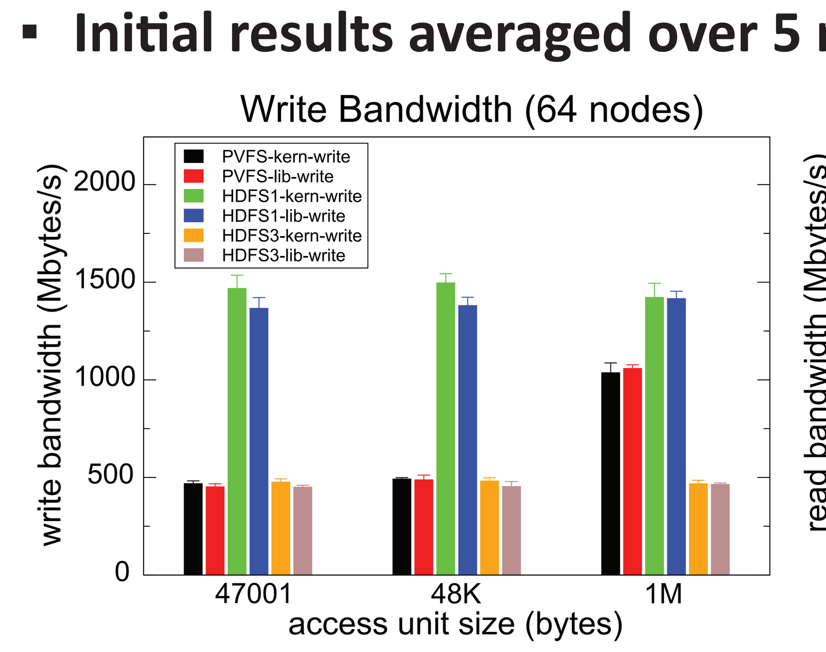
- FUSE or MPI-based filesystem that converts N-1 checkpointing to N-N checkpointing by breaking each node's write operations out into a log file
  - Improves HPC checkpoint performance by avoiding underlying filesystem bottlenecks
  - PLFS's log structured writes fit the filesystem semantics provided by the HDFS cloud storage system
- If PLFS could write its logs to HDFS, it could provide N-1 checkpoint semantics for HPC applications using HDFS for storage

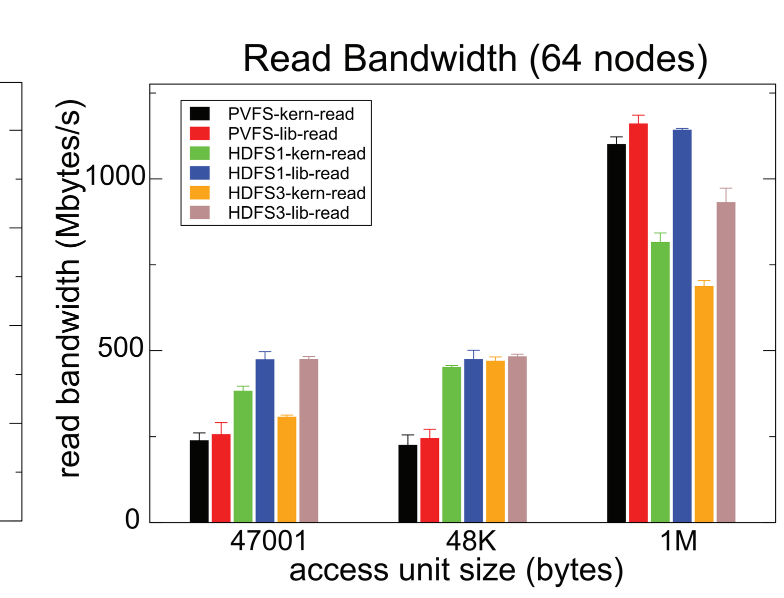


RESULTS

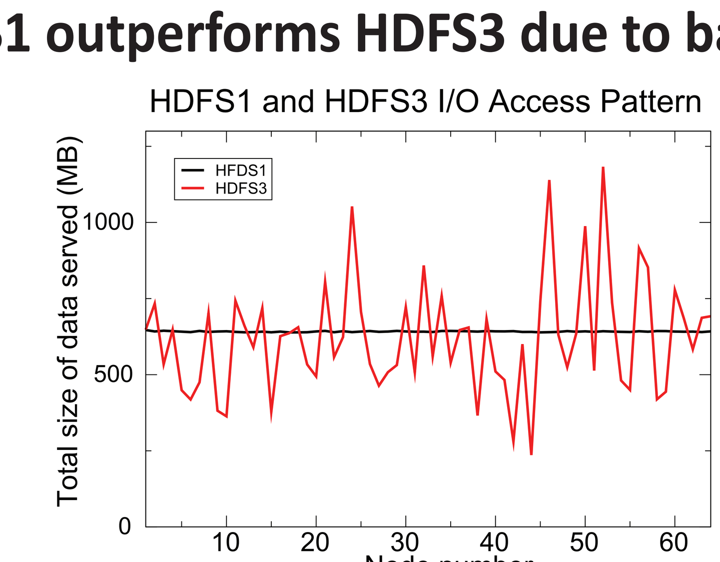
Platform: Marmot PRObE cluster

- 1.6GHz AMD Opteron dual processor, 16GB memory, Gigabit Ethernet
- Hadoop HDFS 0.21.0, FUSE 2.8, PLFS, OrangeFS 2.8.4 (PVFS)
- LANL test_fs N-1 checkpoint benchmark with 47001, 48K, or 1M byte objects
- 6 test cases: PVFS, HDFS1 (no replication), HDFS3 (3 way replication) through a kernel mount point and a library API
- Initial results averaged over 5 runs with std. deviation show





- 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



Intel Science & Technology Center for Cloud Computing

Carnegie Mellon University · Georgia Tech · intel · Princeton University · UC Berkeley