PLFS/HDFS: HPC APPLICATIONS ON CLOUD STORAGE

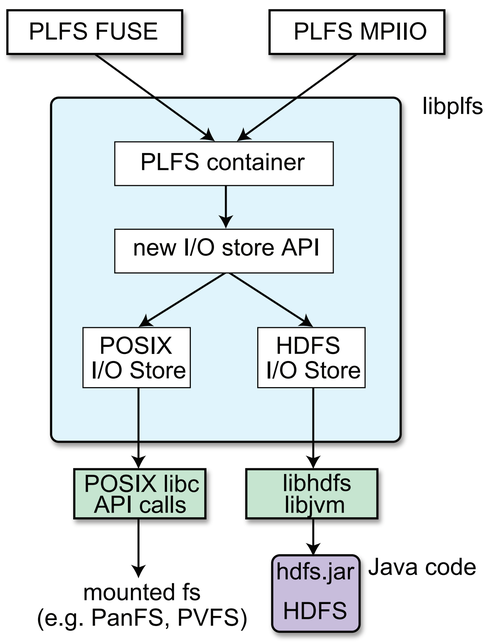
Chuck Cranor, Milo Polte, Garth Gibson (CMU)

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-HDFS ARCHITECTURE

- Insert new I/O store layer into PLFS
  - POSIX I/O store module provides traditional interface to PLFS backing store
  - New HDFS I/O store module uses libhdfs API to store and access logs
    - Hadoop's libhdfs uses Java Native Interface (JNI) to provide C/C++ access to HDFS Java methods
    - Links a Java Virtual Machine into PLFS

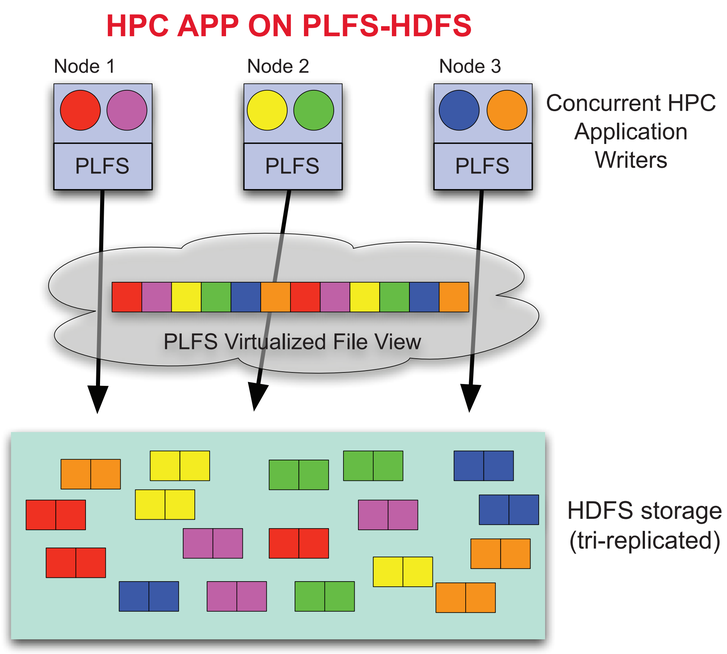


PLATFORM ISSUES

- libhdfs does not work in a child process after a fork
- PLFS creates new threads for each I/O operation, this causes a Java memory leak
- HDFS concurrency bugs exposed by using it in new ways
- 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



API ISSUES

- Must map PLFS I/O calls to either POSIX or HDFS I/O store modules, 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)

PRELIMINARY PERFORMANCE RESULTS

- Extra overhead of Java vs. HDFS optimizations
- Initial testing: 5 node cluster on 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 byte objects
- Will scale up size of benchmark once more Marmot nodes are ready
- 4 test cases:
  - PVFS alone, using the Linux kernel module
  - PLFS over PVFS
  - PLFS over HDFS with 3 way replication
  - PLFS over HDFS without replication

platform	read	write	
PVFS	73.7	94.9	Mbytes/s
PLFS-on-PVFS	63.0	65.7	Mbytes/s
PLFS-on-HDFS			
replication=1	281	242	Mbytes/s
perlication=3	76.8	71.9	Mbytes/s

Intel Science & Technology Center for Cloud Computing

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