POSIX Order Write Test Report

Albert Cheng

2013-11-27

http://www.HDFGroup.org

1. Purpose

The report shows the result of the POSIX Write Order test in different operating systems using different file systems. Section 2 shows the requirements of the test. Section 3 shows the implementation design of the test. Section 4 shows the results of running the test in different operating system with different file systems. The last section is a summary of the results.

2. Requirements

The write order test should verify that the write order is strictly consistent. The SWMR feature requires that the order of write is strictly consistent. The SWMR updates to the data structures in the file are essentially implementing a "lock-free" or "wait-free" algorithm in updating the data structure on disk. See: https://en.wikipedia.org/wiki/Non-blocking_synchronization, for example. In those algorithms, the order of updates to the data structure is critical and if it doesn't occur correctly, the reader can get inconsistent results.

"Strict consistency in computer science is the most stringent consistency model. It says that a read operation has to return the result of the latest write operation which occurred on that data item."--(http://en.wikipedia.org/wiki/Linearizability#Definition_of_linearizability).

This is also an alternative form of what POSIX write require that after a write operation has returned success, all reads issued afterward should get the same data the write has written.

3. Design of Implementation

The test named as twriteorder, simulates what SWMR does by writing chained blocks and see if they can be read back correctly.
There is a writer process and multiple reader processes.
The file is divided into 2KB partitions. Then the writer writes 1 chained block, each of 1KB big, in each partition after the first partition.
Each chained block has this structure:

- Byte 0-3: offset address of its child block. The last child uses 0 as NULL.
- Byte 4-1023: some artificial data.
- The child block address of Block 1 is NULL (0).
- The child block address of Block 2 is the offset address of Block 1.
- The child block address of Block n is the offset address of Block n-1.

After all n blocks are written, the offset address of Block n is written to the offset 0 of the first partition (Block 1). Therefore, by the time the offset address of Block n is written to this position, all n chain-linked blocks have been written.

The other reader processes will try to read the address value at the offset 0. The value is initially NULL(0). When it changes to non-zero, it signifies the writer process has written all the chain-link blocks and they are ready for the reader processes to access.
If the system, in which the writer and reader processes run, adhere to the order of write, the readers will always get all chain-linked blocks correctly. If the order of write is not maintained, some reader processes may found unexpected block data.

4. Results

4.1. Linux hosts with local filesystem

The machines have ext3 and ext4 local filesystems. The test with both write and reader ran in the same host, passed all runs up to 1,000,000 linked blocks, resulting in datafiles ~2GB big.

4.2. Linux hosts with NFS filesystem

The machines run Linux operating system with one using CentOS 5 and the other using CentOS 6. Both hosts access a common NFS filesystem served by a NFS file server, therefore all file accesses are via the network. The test is run with the writer and the reader running in separated hosts. All tests passed with up to 1,000,000 linked blocks, resulting in datafiles ~2GB big. There is a twist—the first run when 500,000 and 1,000,000 linked blocks are used, the reader would encounter failure. But all subsequent runs with the same number of linked blocks would pass without failure. No explanation of this behavior is available yet.

4.3. AIX hosts with GPFS

The machines run AIX 5.3 OS and are as a box of “blades”. All blades share the access to the GPFS filesystem.
The test, including both writer and reader, running in the same or separated “blades”, passed all runs up to 1,000,000 linked blocks, resulting in datafiles ~2GB big.

A side note: when in separated blades, the write time is 7.6 seconds but the reader time is 69 seconds. It is speculated that the GPFS system may be doing some “kernel” buffering to make write time to appear smaller.

4.4. Linux hosts with GPFS

The above mentioned AIX system site also has Linux hosts that share the same GPFS system. The Linux hosts are 64bits system.
The POSIX write order test is run in separated Linux machines using the same GPFS file system. The test, including both write and reader, in the same or separated hosts, passed all tests up to 1,000,000 linked blocks which resulted in approximately 2GB size file.

A side note: the writer took 8.5 sec to write the 2GB file but the reader took 83 seconds to read them. Write speed is 10 times faster than read speed--Kernel memory is in play here. Nevertheless, IBM GPFS adheres to the write order correctly.

4.5. Linux Cluster with Lustre file system

The POSIX write order test is run in a remote Linux Cluster with a Lustre file system. The test passed with small size files such as 200MB in size. But when larger number of linked blocks (e.g. –n 500000 => 500,000 linked blocks resulting in file size about 900MB) and the writer and reader were in separated machines, the reader detected errors in data it read back. The exact cause of the failure is not known yet but it fails for bigger file sizes over separated machines.

5. Summary

This is a summary of the test results in different system using different the GPFS or Lustre file system.

System	Same host/machine	Separated hosts/machines
Linux with local file system	Passed in all file sizes up to 2GB.	N/A
Linux with NFS file system	N/A	Passed in all file sizes up to 2GB.
AIX with GPFS	Passed in all file sizes up to 2GB.	Passed in all file sizes up to 2GB.
Linux with GPFS	Passed in all file sizes up to 2GB.	Passed in all file sizes up to 2GB.
Linux with Lustre	N/A	Passed in small file sizes but failed with large files (e.g., 900MB)