New Patch for MPI

PLFS 2.2 has changed some of the ad_* code such that MPI implementations will need to be patched with PLFS 2.2's ADIO interface in order to allow MPI/IO jobs to use PLFS file systems.

LANL PLFS Purger Fixed

LANL's purger was not working correctly with respect to handling PLFS file systems. It was purging parts of PLFS tree structures by going through the backends and purging out PanFS files that were parts of PLFS files. The purger is now fixed and will not traverse PLFS backend directories, but will purge PLFS files through the PLFS FUSE mount points, as it should.

Error Copying Large File Merges Contiguous Index Entries

Tickets #7 & #57: This problem was noticed copying a ~340 GB restart dump file from a non-PLFS file system to a PLFS file system. It resulted in an error exit code 134 opening the file to read. This was because the PLFS index file was larger than the memory available to hold it. There was an outstanding ticket (#7) to merge contiguous index entries for scalability. This was the answer to this problem. An index compression algorithm was implemented so that contiguous index entries are merged by O(1K). The compression factor can be changed as files get orders of magnitude larger.

Fixed Misleading Variable Names in WriteFile.cpp

Ticket #19: PLFS has the concept of logical and physical files. Some of the variables in WriteFile.cpp were misleading about what was logical and what was physical. This made the code confusing to developers. The merge from EMC engineering that added a LogicalFD class and moved all the code into ContainerFD and added FlatFileFD cleaned all this up. Now only the expandPath code ever sees a logical path. All the internal container and flatfile code and index code (which is part of container) see only physical paths.

PLFS Tools Indicate Version

Tickets #28 & #42: There are a variety of utilities that come with PLFS. Each now takes a "-version" flag that will return the version of PLFS with which they were distributed. The PLFS library also has a function that returns the version for applications that use the PLFS API directly.

plfs_query Made More User-Friendly

Tickets #32, #52, & #55: This utility is intended primarily for system administration and PLFS development personnel. It enables one to learn more about the backend files that comprise a PLFS file or figure out to which PLFS file one of the PLFS mount point backend files belongs.

Fixed Code to Eliminate Comparison Warnings

Ticket #37: There were several warnings about comparisons involving signed and unsigned values. Many of them were comparing to see if an unsigned integer was less than zero (impossible). These were all fixed.

Include Directive in PLFSRC Resulted in Default Values Being Used

Tickets #41 & #44: PLFS verision 2.1 introduced the ability to include files as part of the PLFSRC file. A bug that reset required parameters to default values every time a new file that was included was parsed was fixed. This was most noticeable in that it forced include files to be put in a certain order or values, specifically num_hostdirs, were changed back to incorrect default values. The PLFSRC file parser now expands all include files and ensures that resulting PLFSRC file, as a whole, is correct.

plfs_check_config Makes Non-Existent Directories

Ticket #43: This utility is intended primarily for system administration and PLFS development personnel. It enables one to check if the PLFS configuration defined in the PLFSRC file is valid. The plfs_check_config utility will tell the user when backend directories don't exist for a mount point. Using the "-mkdir" switch will create those backends that don't exist.

Race Condition for N-1 I/O through FUSE

Ticket #45: N-1 I/O through FUSE jobs would fail because the file could not be found. This was a race condition between creation and when the file was accessed. All tests that reproduced this error now work correctly. The PLFS team does not recommend this I/O pattern through FUSE because it is a slow pattern with additional overhead by FUSE. However, it now works for completeness.

Rewinddir Fixed

Ticket #47: The POSIX call, rewinddir, could not be used on PLFS. Instead, the user had to closedir then opendir again. It turned out to be a C macro that was the only line in an "if" statement's "then" clause. No curly braces, {}, were used because it was one line. However, the macro expanded to two statements and so the second statement was always executed, even when it should not have been executed. We've changed our coding standard to ask developers to always use curly braces, {}, irrespective of the number of statements that follow (for loops, ifs, etc.).

ADIO Interface Patched for Cray

Tickets #48 & 56: Fixed the ADIO interface and add some patches so that it can be compiled out of the PLFS repository for use with Cray's xt-mpich2 MPI without Cray having to make changes. Furthermore, all functions have either "plfs_" or "adplfs_", depending on where they are defined, prepended to them so that they don't conflict with other libraries.

Better Logging Capability Implemented

Ticket #53: PLFS used a custom plfs_debug logging capability. This was transitioned to a more flexible one based on “mlog”. Information on how to configure the mlog-based logging is in README.mlog.

plfs_init Ensures “init” Routines Called One Time

Ticket #59: There were several things that happened when PLFS gets initialized. For a while some things were getting done more than one time and that caused problems. The plfs_init routine has been fixed so that all the initializations are done one time and there are no unintended side-effects to multiple initializations.

Error in ANL Test noncontig.c

Ticket #60: This bug was caused due to opening the file in Read-Write mode w/ truncate. There was a bug with trunc of an open file. Initially the bug was that metadata droppings wouldn't rename properly, but this unveiled a separate bug where physical offsets wouldn't get correctly updated. These are both fixed. The PLFS team recommends against using O_RDWR. Please use O_RDONLY or O_WRONLY. There are performance penalties for using O_RDWR.

“Flat File” Mode for Better N-N Workload Performance

Ticket #64: The "Flat File" concept was developed and implemented for the N-N workload. Any one file is written contiguously to a PLFS backend. Multiple files can be written concurrently when a PLFS mount point has multiple backends. This provides parallelism that is transparent to the user. There are no indexes, containers, metalinks, etc. Thus, the PLFS overhead of turning the N-1 pattern into N-N is eliminated. A mount point cannot support both "Flat File" and "Container" mode. The same backend directory specification can't be used in both "Flat File" and "Container" mode. A site must handle this administratively so that the mount points are properly declared to meet these criteria.

Truncating an Open File Doesn’t Work

Ticket #66: The metadata for a file that was truncated wasn’t being updated. So, the file had the new contents, but the old length. That made a user of the file get the wrong size for the file and use it incorrectly. All metadata is properly updated now so that the size of the file is correct after it is truncated.

PLFS Build within a PLFS Mount Point

Ticket #77: There was an error in the “./configure” script run that indicated a problem building PLFS within a PLFS mount point managed by PLFS 2.2rc2. A metalink was not resolving to the correct physical location. This bug was fixed and PLFS can now be correctly configured and built within a PLFS mount point.

Overwriting N-1 File with Less Data Shows Wrong Size

Ticket #78: When you overwrite a N-1 PLFS file with less data than it originally had, it does not show the correct size. However, if you write it with more data, it does show the correct size. There was an inconsistency in how the metadata information was maintained. It was fixed and PLFS now reports the correct file size in all cases.