PLFS Versioning Plan

1 Preface

This document attempts to lay out a design path for the planned development versions of PLFS. It also attempts to describe how development versions can move into production and the various issues for transitioning between versions. Some of the issues are technical and some are policy. Some of the technical issues are things like how do we implement directory management in the future metadata distribution layer. Policy issues are questions like whether incompatible layouts are handled transparently by PLFS or whether they are handled by explicitly copying data from an old PLFS into the new one.

Our goal is to get PLFS into production at LANL. However, we want to make sure that the production releases are bug-free, give performance attractive to users, and are as easy to use as possible. Hopefully by spelling out the issues involved the current version of PLFS and the planned versions, we can identify the target version for our first production release. There is of course a tension between releasing something soon and releasing something good. We could release the current version today but it has several limitations that will be discussed below and we need to determine whether we are comfortable with those limitations.

At the end of this document, we will also describe current open issues and other miscellaneous development plans.

2 Current Version - Version 0

The current version has support for both PLFS-FUSE and for PLFS-ADIO. However, the PLFS-FUSE implementation transparently maps logical paths into physical ones. It knows this mapping via a single mount argument naming the physical back-end store. As is true for PanFS access, each user's PLFS back-end store is located on a different PanFS volume. This is accomplished via symbolic links. For example, */plfs/ben* is a symbolic link to */mnt/plfs/vol5/ben/.plfs_store*. When PLFS is mounted, it is mounted on */mnt/plfs* and directed to use */panfs/scratch* as its back-end store. So a typical name resolution would work as follows:

1. user ben opens logical file */plfs/ben/foo*
2. the sym link resolves this to logical path */mnt/plfs/vol5/ben/.plfs_store*
3. FUSE strips the */mnt/plfs* and PLFS inserts the back-end store path and resolves to physical path */panfs/scratch/vol5/ben/.plfs_store*

This works great for PLFS-FUSE. The back-end store is hidden from the user. However, for PLFS-ADIO, there is no current support for mapping so when a user wants to use PLFS-ADIO, they need to know and use the path to their back-end store. We have observed that users are confused by the notion of the back-end store and therefore we believe that the PLFS-ADIO in the current version is not ready for a production release.

We can have a production release with just PLFS-FUSE, but Meghan has observed much higher bandwidth with the PLFS-ADIO layer and believes that the bandwidth with just the PLFS-FUSE layer might not be enough to convince users to switch. Recently, the developer of Bulk-IO has made some optimization by tuning the IO to match the underlying stripe configuration of PanFS and although the performance of PLFS-FUSE is better than Bulk-IO it is not several times better whereas the performance of PLFS-ADIO is several times better.

3 Version 1

Version 1 will move the mapping logic from PLFS-FUSE into the library so it will be shared with PLFS-ADIO. However, we can no longer rely on the mount argument to declare the back-end store as that is not seen by PLFS-ADIO. We could use the mount argument for PLFS-FUSE and use hints for PLFS-ADIO but this exposes the

back-end store to users and we have already decided we don't want to do that. Therefore, we have decided that we will use an */etc/plfsrc* file on every node to declare mappings between logical paths and back-end stores. This file will be parsed by both PLFS-FUSE and PLFS-ADIO.

However, one problem is that we can no longer depend on symbolic links to spread users across volumes. We can either put all back-end stores on a single volume or we can put a mapping entry into */etc/plfsrc* for every user (*e.g. /plfs/ben* maps to */panfs/scratch/vol5/ben/.plfs_store*). This might be a maintenance problem however as every time a new user is added, every */etc/plfsrc* file will need to be updated. I don't know the administrative cost of this.

3.1 Compatibility

Version 1 will be backwards compatible with Version 0.

4 Version 2

A better way to distribute users across volumes is for the */etc/plfsrc*

5 Open issues and miscellaneous plans

5.1 Container identification

Container identification is a pain in the butt. SUID doesn't really work and now *container/accessfile* also seems not to be working.

5.2 PLFS-ADIO optimizations