

Google Summer of Code 2006

Application proposal

Name: Ricardo Manuel da Silva Correia

E-mail Address: rcorreia@wizy.org

Project Title: ZFS filesystem for FUSE/Linux

Project Goals:

The goal of the project is to implement the [ZFS filesystem](#) on the [FUSE](#) (Filesystem in Userspace) framework in the Linux platform. This project will be implemented in the C language and will be licensed in the CDDL (OSI-approved) license.

ZFS is a modern advanced filesystem from Sun.

Project Benefits:

This project will make it possible for Linux systems to be more interoperable with Solaris/OpenSolaris. The Linux operating system has a very large installed base, and if ZFS was available to these users, it would ease the transition to Solaris/OpenSolaris, if it proved advantageous. DragonFlyBSD and Apple have already announced plans to implement ZFS (for Mac OS X, in the case of Apple), so future interoperability with these systems is expected. After this project is completed, it will also be easy to use ZFS filesystems in FreeBSD - since it already has an implementation of FUSE, it will be relatively easy to port to it. It will also be possible to port this to FUSE/OpenSolaris, although the usefulness of that is debatable :)

Project Tasks and Deliverables:

Having in mind the [suggestions of ZFS's engineers](#), this project will be implemented in the following phases:

Phase 1 – Thorough study of the ZFS source code and on-disk structure. Determination of the preliminary structure of the project code/modules.

Phase 2 – Porting of *libzpool*, *ztest* and *zdb*. According to the [“Porting ZFS”](#) page, *libzpool* is a userland port of the SPA (Storage Pool Allocator) and the DMU (Data Management Unit) which consists of about 80% of the ZFS kernel code.

Phase 3 – Porting of the *zpool* and *zfs* tools. The OpenSolaris ZFS userland <-> kernel interface is currently done with *ioctl*s through a device file (*/dev/zfs*). Naturally, as this will be a userland filesystem implementation, a new approach is required - the best solution seems to be through a UNIX socket. After this phase is complete, it will be possible to work with and create *zpools*.

Phase 4 – Implementation of the ZPL (ZFS Posix Layer) through FUSE. This will have to be implemented from scratch. After this step is complete it will be possible to mount and use ZFS filesystems.

[Note: *zvols* (ZFS volumes), given its arguably limited usefulness, will be implemented only if the effort required will not cause delays. Otherwise, it will only be implemented after the completion of the project.]

The final outcome of the project consists in a userspace program that mounts a *zpool* and all the filesystems inside it in a user-specified directory. The project will also contain the tools necessary to create, change properties and delete filesystems and *zpools* (both in its simple form and with stripes, mirrors and/or raid-z), create and delete snapshots and clones, as well as check the integrity of all the

data inside a *zpool* (scrubbing). Both files and block devices will be able to be used as *vdevs* (virtual devices).

Project schedule:

Until the end of the (school) semester I can work on this project as if it was a part-time job. In the summer, I can work on it full-time. This is my proposed time schedule. This is only an estimate:

Phase 1 – 1 week

Phase 2 – 4 weeks

Phase 3 – 1 week

Phase 4 – 4 weeks

If the project goes according to schedule, I'll use the 2 remaining weeks for thorough QA and performance optimization.

Project limitations:

The project will have the following inherent limitations:

- It will be a userspace filesystem. Due to context switches and other limitations, performance can never be as good as a kernel-level implementation. However I will make an effort to make it as efficient as possible (given the time-frame).
- File locking will not work. This is a current limitation of FUSE, so there's nothing I can do about it.
- No support for writable shared memory maps. Again, this is a current limitation of FUSE.
- The license will have to be CDDL. This is because I intend to use code from the OpenSolaris ZFS implementation (which is CDDL-licensed), both in the spirit of open-source and code reuse, and because of the limited time-frame.

Questions:

1) by Diego Pettenò:

Q: I'm not sure about a possible license clash between CDDL and GPL/LGPL in which FUSE is released. Was that considered already?

A: Yes, I've considered it. Quoting from the [FUSE FAQ](#): *“Under what conditions may I distribute a filesystem which uses libfuse?” ... “In simple terms as long as you are linking dynamically (the default) there are no limitations on linking with libfuse. For example you may distribute the filesystem itself in binary form, without source code, under any propriatery license.”*

Personal statement:

I'm currently studying Computer Engineering in Portugal and I'm a software engineer by heart. I've been programming computers since the age of 8, and I'm a long time Linux user. Only recently I've become interested in OpenSolaris and NexentaOS, mostly because of the ZFS filesystem and DTrace. I'm fairly confident I have the skills necessary to implement this project, and I'm very enthusiastic about it (I've been meaning to do it since I've read about ZFS).

After Google Summer of Code 2006 ends, I intend to continue to maintain and improve this project, and when it's considered fully stable, I intend to use it myself in most of my disks (including my root filesystem). I also have some ideas for additional features/tools, but that will have to wait ;)