

Tuesday, January 8, 2008

## SamFS at home?

Most people think, that Hierarchical Storage Management is one of the technologies without a use at home. Thus it's not a product many people know. It's something for big companies, to store vast amounts of data. But yesterday an idea struck me: This isn't the case. You want SamFS (or the "hierarchical" functionality from it in another filesystem, e.g. ZFS). You really want it, albeit SamFS need some rather small extensions to make it a really cool solution.

### The problem

I've described the problem before. With the rise of high megapixel cameras, digital video, set top boxes with recording the data in the normal household starts to get really huge. Some data is not so important, for example the recording of the last House M.D. episode. Some other data is absolutely mission critical: Try to explain your significant other, why and how you lost all the photos from the last vacation in New Zealand. Some data is so important, that you can't even afford to loose it in the case of a burning roof. You have different needs for different data. This sounds familiar to people who worked in the past with hierarchical storage management.

### SamFS and some technical basics

Sun has such an HSM system. It's called SamFS. And my considerations in the next paragraphs are based on my experiences with this software. At start, SamFS is a filesystem. It's fully posix-compliant, thus all other functionalities are transparent to the users or application. The added benefit of SamFS is the cycle of archive, release and stage. As an example: You put a file into a SamFS filesystem and close it. The archiver steps in and moves it onto different archive media, for example a disk drive, a tape or a magneto optical disk (as you define it). You define policies for it: I want to copies (one on disk, one on tape. In front of this you have additional disk storage. It's called Cache. You can afford to loose this disk storage, as you can recover all data from other media, as soon as the archiver did it's job. The cache has the sole purpose to accelerate accesses to the other media. This is not so important for disk storage but for a tape technology that needs a minute or so to get the first bytes streaming this really helps to speed up things. As long, the data is in the cache every access to the filesystem acts as a normal filesystem. The requests are served out of the cache.

This cache isn't endless. You can size it to same amount of your data. But you don't have to. An interesting phenomenen comes to help: 80% of the time you need 20% of the data, and much data is accessed once in one or two years. The rest just rotates with the rust. It's the same phenomenon i observed when i lived at my parents: 80% of the time you use 20% of the dinnerware. Then there is the dinner ware for family parties and at last the ugly dinnerware the mother-in-law made as a gift. You use it only in the case of an announced visit of the mother-in-law. Your daily dishes are in the kitchen, but you kitchen hasn't endless space. So your sunday dishes are in the living room and the "mother-in-law"-dishes in the deepest corner of your cellar. Data is so similar. With SamFS you can define policies, which media is used.

Okay, back to the cache: So when your cache fills up to a certain amount, you have to act. Take into consideration, that all you data has been archived to other media. You can afford to delete some data. The release cycle of SamFS does exactly this task: It deletes data automatically from the cache based on policies: For example delete all data beginning with the least recently used file until 10% of the cache is free again.

At some time in the future you need your data back. Your significant other want to see some obscure photos of a boring cave you've visited in an other vacation, or to combine the examples: The photos of the birthday of your mother-on-law. Normally you would search for the DVDs and mumble some nasty curses to the gods of data retrieval. SamFS makes this task easier. It doesn't delete the data completly from the cache. It leaves some meta data on it, for example all the posix attributes and where the data was archived. When you access the data, the filesystem retrieves the data from the archival media and delivers it to the applicaiton or user. The user or the application doesn't notice this step besides it may take some seconds or minutes. This is the staging part of the cycle.

### How can you use it at home

So: Albeit the data volume may be different to the needs in the enterprise, the usage patterns are similar. Thus the same tools can come to a help. You don't have to make backups anymore, as doing archive copies are inherent to the process and you don't have to hassle around with more and more rubbish on you relatively small harddisk on your notebook.

SamFS was designed with large tape libraries in mind. So i come to an idea to extend SamFS to make this a fscking cool

## Blog Export: c0t0d0s0.org, <http://www.c0t0d0s0.org/>

tool for storage needs at home. We need a protocol and means in SamFS to access an network based data storage service called via XMLRPC calls via HTTP via SSL. Think about a webservice that understands three commands: GET/PUT/DELETE (sounds familiar? ). Instead of defining a tape library as an archival media, you define this networked storage service.

There are several provides of managed storage out there. The S3 Service of Amazon is just one, albeit the most widely known service. I hope you got the idea already. Imagine a proxy application that translates between the simple protocol in direction to SamFS and the more complex one in direction to the storage providers. Add some encryption and compression to it and you can use any storage provider to archive your data.

The nice thing: You don't have to think about it. You don't have to backup it. You don't have to recover your data. You only have to reinstall the operating system and get the actual copy of the SamFS meta data. The filesystem recovers while using it, bootstrap recovery when you want it call this way.

The proposed extension has two advantages: It would broaden the user group for SamFS and it would give an automatic way (nothing manual happens regular) to do archival at home. It would solve some of my urgent problems as a semi-pro photographer for example. And i'm sure it would solve similar problems for other people.

Posted by Joerg Moellenkamp at 13:44

I don't know what backends samfs supports, but if it accepts any standard filesystem, it might be able to use a fuse-based xmlrpc filesystem. Not as good as a direct support in samfs, but it could help.  
Anonymous on Jan 8 2008, 16:45

When you mentioned this the first angle I thought of was to have one disk (or a mirror) spun up (or maybe flash...) and allow a few terabyte disks to spin down most of the time.

The web service idea is way cool. Next on the list would be "family mode" so that multiple households could share the same backend for things that they would like to share.  
Anonymous on Jan 8 2008, 18:05

One thing that's implicit in this idea - and I think it's a really neat idea - is that you'd really have to trust your remote managed storage provider not to lose - or disappear with - your data.

(Unless, of course, you use two storage services in parallel...)  
Anonymous on Jan 8 2008, 18:40

Yes, the SamFS cache should keep most requests off the normal hard disk, thus the power management of the disks should be able to kick in.  
Anonymous on Jan 8 2008, 18:48

You are correct. I would use two providers (or one local and one remote copy). I think, i will file this as a RfE as soon as possible for SamFS.  
Anonymous on Jan 8 2008, 20:25

Is SamFS free already (or priced in a way that normal home user could afford?)

For the HSM-at-home idea, it would be also nice to have a SamFS extension that burns data to DVD-R (or something like Blu-Ray) and in the retrieval case: shows a popup window on the screen that requests me to go and insert the correct DVD.  
Anonymous on Jan 8 2008, 22:03

Great idea. SGI had this already with their Data Migration Facility. We would specify both tape and FTP Media Specific Processes. The FTP-MSP was used to transfer files to our HPC center, where the giant STK silos are kept. We also conveniently wrapped the transfer with stunnel, for encryption (I think Blowfish was used, as it slowed us down the least).

SAMFS is quite a bit like DMF. It's just a shame what Belluzzo did to SGI...

We just began testing SAMFS, and I'm excited about the potential. If I could only get our brand new SL500/HPLTO4 combination to work!  
Anonymous on Jan 13 2008, 04:14

I think the rfe should be directed to the opensolaris project automatic data migration <http://www.opensolaris.org/os/project/adm/> sounds like it will solve all the problems.  
Anonymous on Feb 15 2008, 21:03