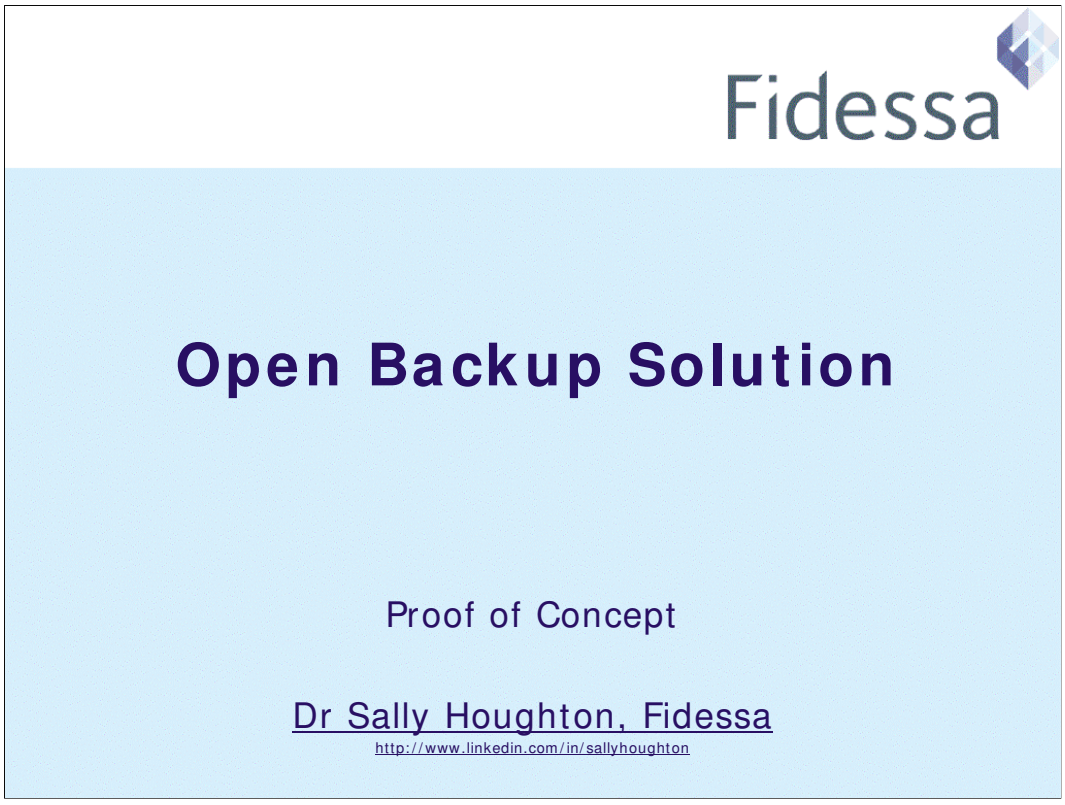
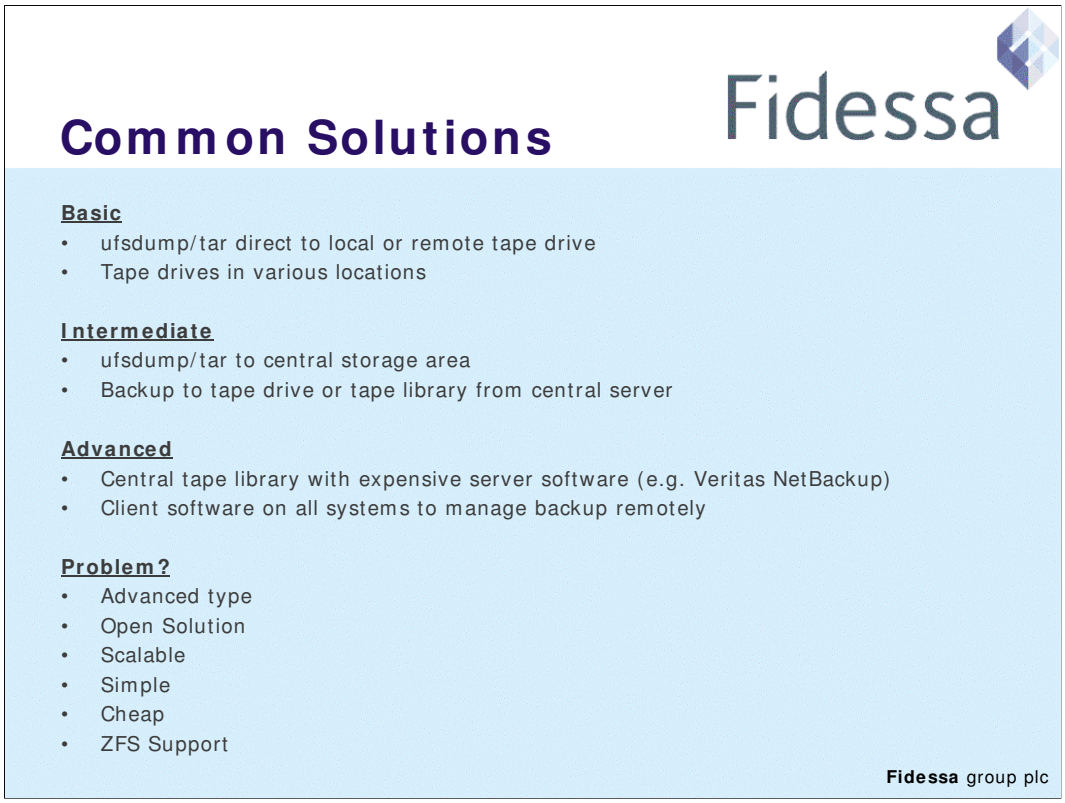


This presentation will be discussing an open backup solution that we have been progressing through a proof of concept at Fidessa.

I will be covering

1. why anyone might want a new backup solution
2. the architecture of our proposed solution
3. the results so far from our testing
4. and, finally, I will go over how much it should all cost



Current backup solutions generally fall into three main areas – I'm going to call them Basic, Intermediate and Advanced.

These vary depending on the size of environment you have and how much money you want to spend.

Starting at the basic level, which I would hope all of us have had experience with – all that's required are some scripts to backup your files to either a local or remote tape drive – whether you use ufsdump, tar or cpio is up to you.

As you increase your server estate, you usually tend more towards a central backup server of some sort, perhaps even one per datacentre location. You now have your scripts either sending directly to the tape drive/library on that backup server, or better yet, you stage the data to disk and then transfer it to tape at your leisure.

For most enterprise environments, you usually go right up to a central solution with some expensive software on it – Veritas NetBackup, Legato Networker or others. There are backup agents installed on every client system and perhaps on every zone – incurring an extra cost and maintenance for updates.

However, what if you want to get to an advanced solution without the cost. In fact, does your current software solution scale with your growth, does it keep things simple and cheap? How easy is it to get your files back? Do you need to install proprietary software in order to get to your data? Does your current solution support ZFS - now that we all want to go that way? Are you locked in to a particular hardware solution (for example, de-dup'ing devices)?

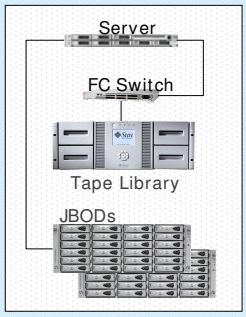
Maybe your company, like Fidessa, prefers to invest in people and not product. There are, after all, many open source products available for experienced sysadmins to use. So we put our heads together and came up with a proposal that's not far from what many others are already doing (I have links on the last slide for similar solutions)…

Proposed Solution

Fidessa

Fidessa Archive & Retrieval Module

(FARM)



Client

- Backup script under application control
- Legacy Support for UFS: `rsync --inplace`
- Migrate to ZFS: `snapshot`, then `send` & `receive`
  - Application back up sooner

FARM

- ZFS hierarchy per Client
- User account per Client
  - rsh/ssh access
  - ZFS delegated admin
  - ZFS snapshot control
- Small/Medium: zpool per JBOD
- Large: zpool striped across JBODs

Fidessa group plc

Our solution defines a FARM - a Fidessa Archive and Retrieval Module.

The FARM at its most basic level consists of a backup server, some disks and a tape library all connected on the network. Depending on requirements, all these pieces can scale up separately – a bigger server, more tape libraries, more and more disk or a dedicated backup network.

The only restriction we're working with here is that the backup server must be able to use ZFS – so it runs OpenSolaris or Solaris 10. Everything else can be whatever you want it to be – commodity hardware. For ZFS, in fact, it's best to have the disks as dumb as possible – no hardware RAID controllers to get in the way.

One of our main desires with the solution is to cut out the middle man. The way we've interpreted this is to say, why should our application support people contact technical support to request a restore. Or why should the backups happen at some time the application support people have no control over. So, from our Client system, we want the backups under the control of the application it's running. It is already a normal practice for application support staff to control when they shutdown or quiesce their application in order to backup. Our solution would give them a script to run when they have shutdown – a script that will perform the backup to the central FARM. Whether they control this manually or via some automatic scheduling software like autosys, control-m or even cron will be entirely up to them.

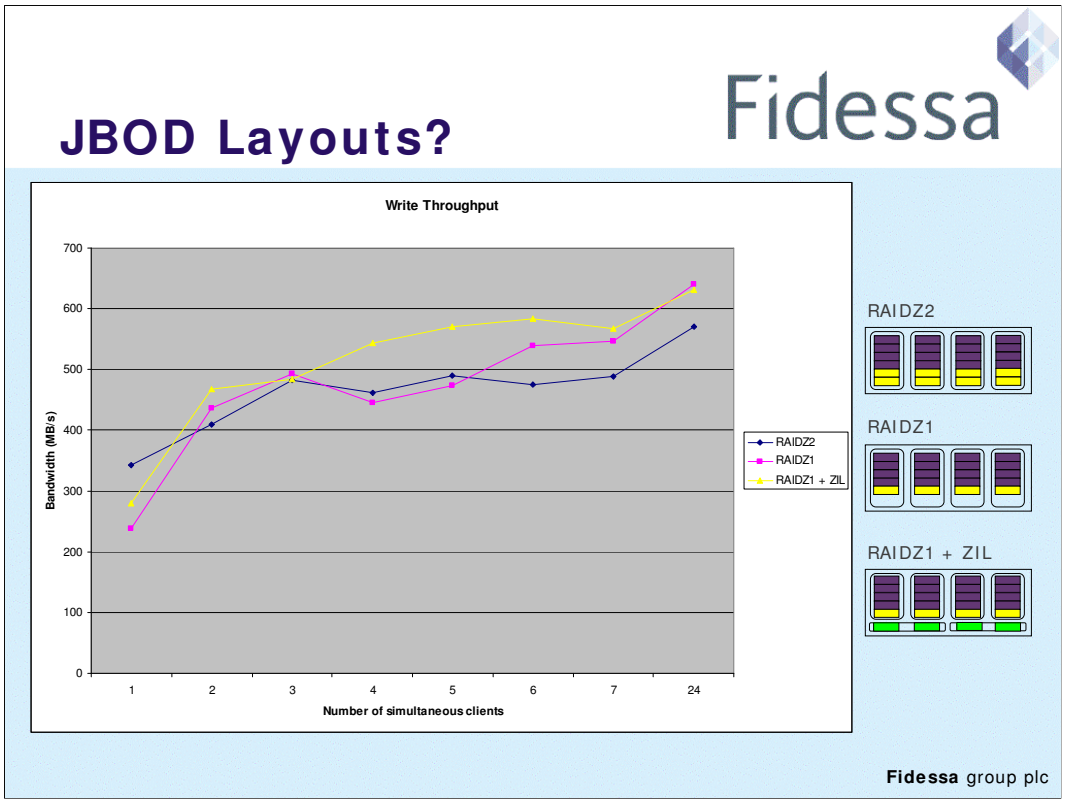
The script will take into account the hopefully legacy UFS filesystem by using rsync and allow us to easily migrate to ZFS. In both cases, we can get block-level incremental backups happening each day and send those over to the FARM. In the case of ZFS, then with snapshots the application can re-start pretty much straight after it's closed for a backup. Then the ZFS send and receive can happen in the background, resulting in better uptime for your application. In fact, the Client could actually be a laptop with Windows and rsync or a Mac using ZFS. This would cover the travelling consultants with no extra license cost.

On the FARM, each Client system will have a hierarchy of ZFS filesystems. Not only that, but each Client system has a standard Unix account on the FARM – this account has delegated admin for all the Client ZFS filesystems giving the Client full control over creating and deleting snapshots without compromising security. In this way, after one initial full sync of data, every day is an incremental backup. This reduces network load without losing the ability to put a full backup to tape.

Not only will this enable us to have continuous incremental backups, but we will also be able to store historical data on the FARM – hopefully enough to ensure that we don't need to go to tape for a restore ever again!

Depending on the amount of data being stored in the FARM, it's possible to either start with 1 or 2 JBODs and then increase as needed. In this case, we believe that you should limit yourself to one pool per JBOD minimum, so that if you ever lose a JBOD, you won't lose all your filesystems. If you know from the outset that you will have a lot of data, then you could just go straight up to a similar solution to Amber road and create your RAIDsets in the zpool as stripes across multiple JBODs. In this way, you improve the redundancy from one/two disks per RAIDset to one/two JBODs (depending on which form of RAID you choose).

As our proof of concept only involved a single JBOD, we limited ourselves to one pool and then we tried different RAID layouts to see how they performed…



Here we have a graph of our measured bandwidth for writes during the rsync process against the number of hosts running simultaneously. It's linear up to 7 simultaneous hosts, but as this was showing a plateau effect, we took the next step as all our 24 test hosts at once.

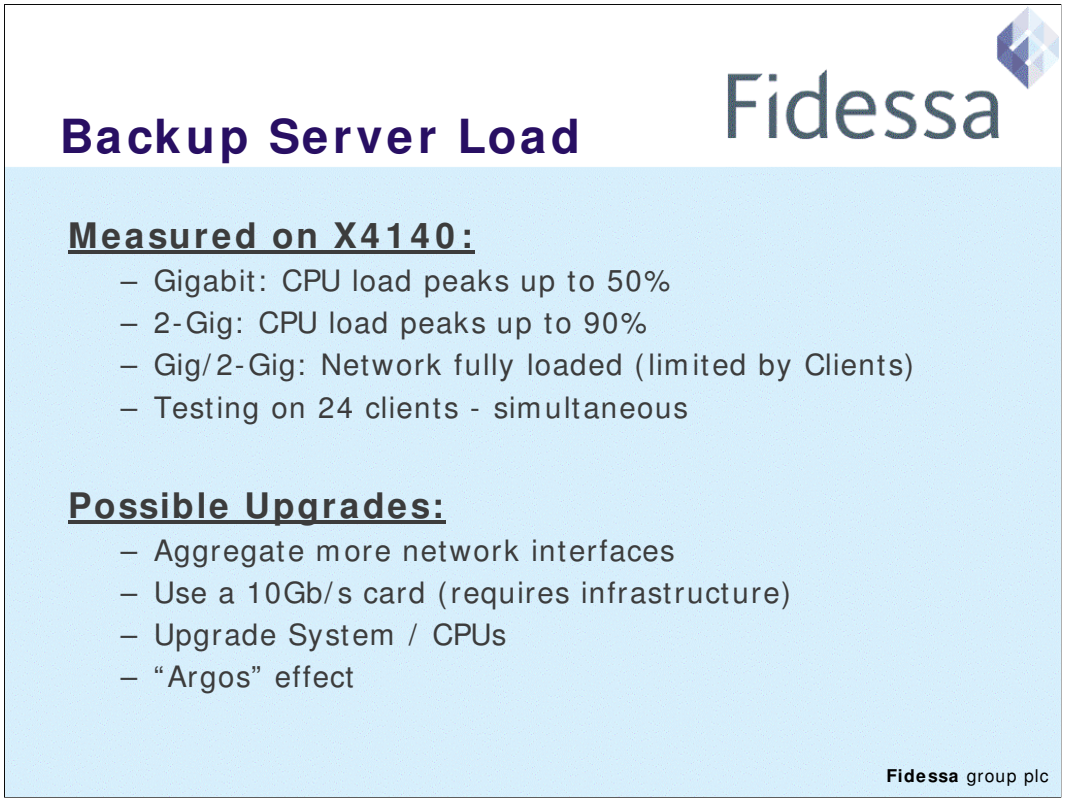
Generally, we don't see too much difference in performance between the three types we chose: RAIDZ2, RAIDZ1, and RAIDZ1 with a dedicated ZFS Intent Log. Although RAIDZ1 with ZIL can usually be seen as slightly better. In fact, it's pretty nice to see even a slight improvement as the disks for the ZIL are the 1TB 7200 rpm disks. Although we haven't tried yet, we would expect even better results if we put in some smaller but faster disks – like 15000 rpm ones.

However, what we really want to do is give ourselves the option of using Solid State disks for the ZIL.

Having just been on the SSD Discovery Day, I am quite excited about getting some of these on try-buy and seeing how they improve our backup window. Clearly, the data still has to be written to disk, but leaving all the control to ZFS means that ZFS can coalesce threads together to provide better performance even when the disks are getting thrashed.

As a simple example, a demonstration on the SSD Day showed a small tar happening on a ZFS filesystem over NFS. With a single 32GB SSD for the ZIL, the time to un-tar the files improved from 1 minute to 9 seconds!

So, let's have a look at the effect this has been having on the backup server in the FARM…



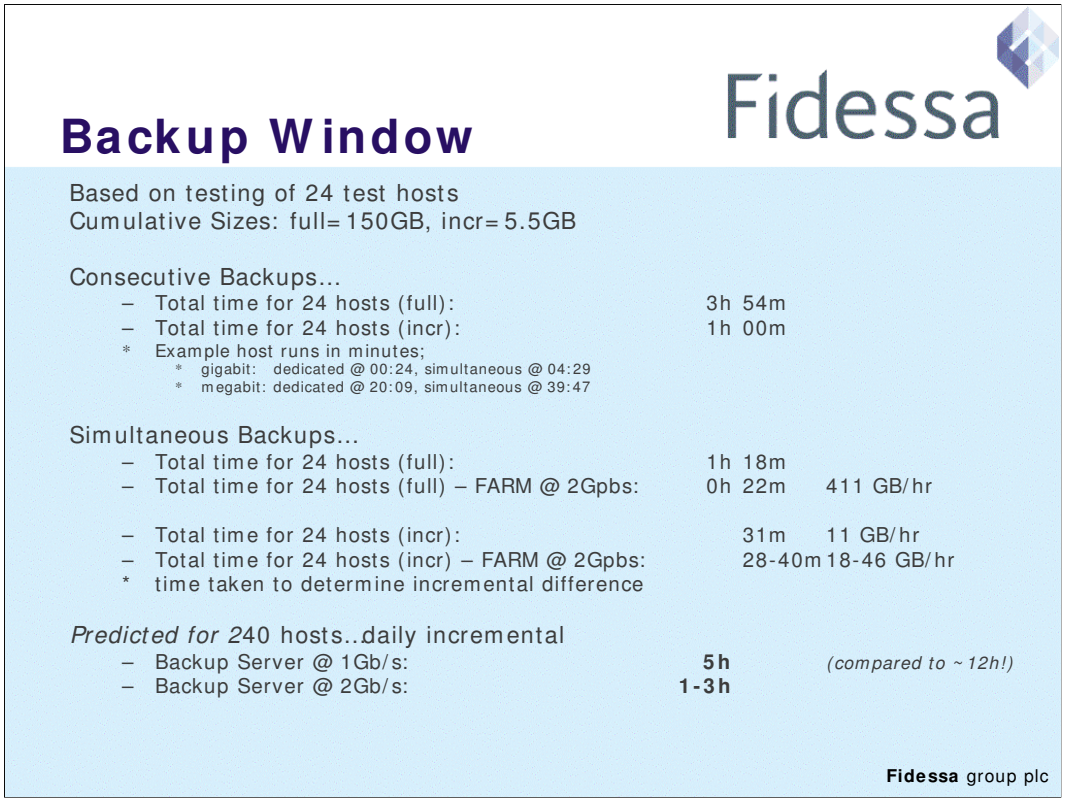
Our backup server for the proof of concept was an X4140 with 2 quad-core CPUs running at 2300MHz. We tried our runs on both a single gigabit connection and on two aggregated connections providing us with 2 Gbps.

The CPU utilisation during the gigabit tests showed that we are only peaking at about 50% CPU usage and for 2-gig we saw a few small peaks up as high as 90%. Understandably, the network bandwidth was up at 100% for most of the time.

Generally, it's not a bad thing to utilise the backup server as much as possible – after all, that's what it's there for. However, for only 24 test clients and an intention of ramping this up to 240 clients, we believe that we may need to upgrade the server to a larger one before we go live with the solution, especially if we want to aggregate more interfaces.

From the testing we've done, we could see no reason to artificially limit the number of simultaneous connections, however, as I've said, we're only testing on 24 clients. One of the possible upgrades we have already started scoping is to script a priority or ticketing system for the client systems…something that we are calling the "Argos" effect. Each client system initially connects to the FARM and gets a ticket number. Then it sits and waits and keeps looking up at the screen to see whether there number is up and they are allowed to go ahead and initiate the backup. As the number of concurrent connections is tuneable from the FARM itself, then this could be easily changed on-the-fly upon need, say if your network fails or gets reduced in capacity or it you want to fast-track a particular client.

Really, it all comes down to running the FARM as fast as possible in order to keep the backup window for the clients manageable…



So, looking at the backup window for our 24 hosts…the numbers we have are based on a full, one-off, rsync backup of 150GB and an incremental size of approximately 5.5 GB for all 24 clients. If we run the host backups one after the other, then the time for all of them to complete a full backup is about 4 hours, but the real measure is the normal, daily incremental which is 1 hour.

If we run all the host backups simultaneously, then we get a saving of about 50% on that with the daily incremental taking about 30 minutes. The statistics per client here aren't that great with some hosts taking considerably longer to sync – for example, one of the gigabit clients if left by itself would take about 24 seconds to do a daily incremental, but when run with all the other hosts it takes 4.5 minutes.

When we tried the 2-gig aggregation, initially, this showed no improvement at all. We quickly realised that this was due to a few, large hosts being on megabit connections – not too surprising as these were test servers and not production. What was happening was that the host network was saturated while the backup server wasn't. So, we moved the top three sized hosts to gigabit and tried again. Not only is this a bit like moving the goal-posts during the game, but the amount of data started changing on the test hosts quite dramatically. In fact, we got up to a cumulative size of the daily incrementals of 22 GB instead of the original 5½ GB. So I've converted the incremental dump times into a measure of GB transferred per hour.

The numbers that we got on the 2-gig tests were much better, not only was the **full** copy of the data onto the FARM down in elapsed time, but we were getting a throughput of anywhere from 18 to 46 GB/hr. The main limiting factor here is rsync – it takes time to determine the changes. As we migrate clients to ZFS, this should improve greatly because ZFS just knows what's changed…unfortunately, this is not something we were in a position to test at the time.

Just to see that we were on the right track, we made some rather bold assumptions about average host data size and bandwidth and assumed that we can scale from the proof of concept up to 240 hosts. This came back with a vast improvement over our current backup window. With the FARM on a gigabit connection, we would be able to complete the backups in less than half the time. And with the FARM running on 2 gig, we could hope to backup all 240 clients in less than 3 hours.

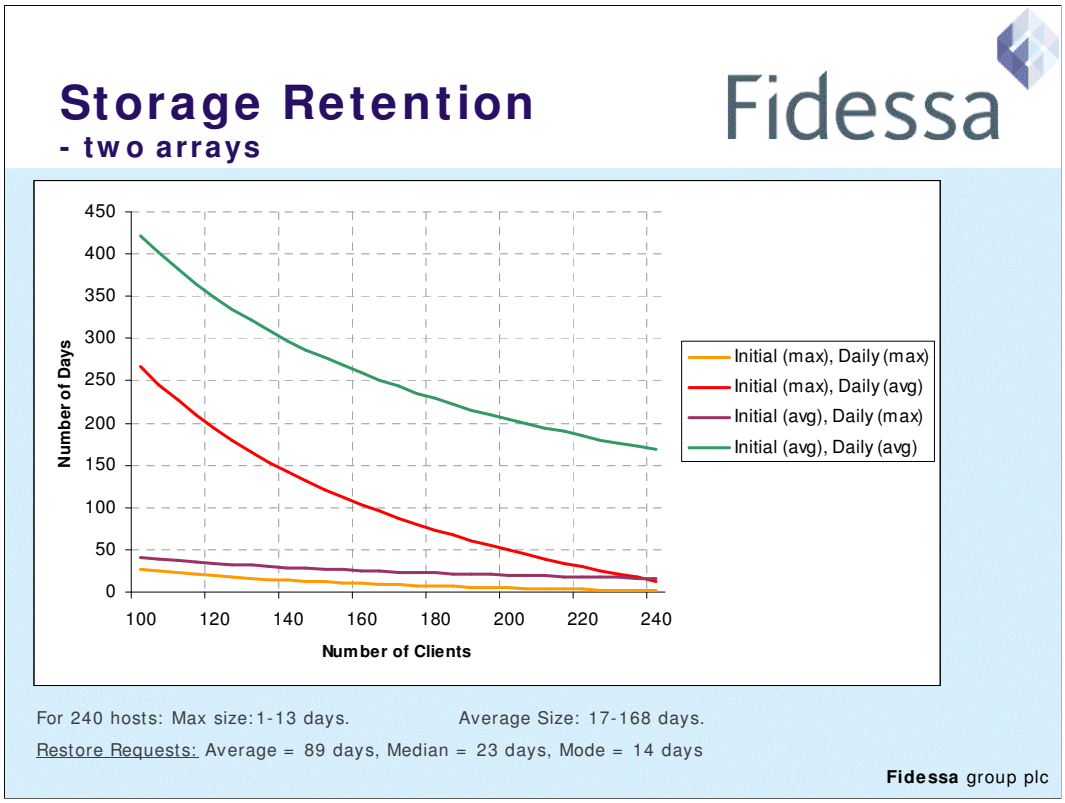
Given these windows and storage amounts, then, we also wanted to know how long we could keep data down on the FARM…

Raw numbers;

100Mbps = 44GB/hr and 1000Mpbs = 439GB/hr and 2000Mpbs = 878GB/hr

Initially, 24 clients; 3 on gigabit, each with 2 zones (i.e.. 6 clients)

After; 24 clients; 6 on gigabit, each with 2 zones (i.e.. 12 clients)



This graph shows 4 different choices depending on whether we use the maximum amount recorded or the average amount recorded for either of the initial full sync or the daily incremental sync. This is all calculated on two J4400 arrays which gives us about 28 TB of storage.

Now if every client system had 114 GB in total size and increased by 8.5 GB each day, we could store them all. On the other hand, if every client system had about 7GB of data and changed by less than a gig each day, we could store 168 days history for each client.

Realistically, about 85% of our clients have an initial backup size of less than 20GB. For an initial maximum size of 20GB with an average daily backup, then the results stay pretty close to the green line and mean that we could store 151 days historical data for 240 hosts. From our results, then, we can say that if the usage of the systems matches those that we've tested on, then we can keep enough data to store approximately 100 days of changes.

With respect to the restore requests that we've received over the last two years, then the majority of restore requests will be able to be serviced from the storage array without having to retrieve remote tapes. This speeds up and simplifies the restore process.

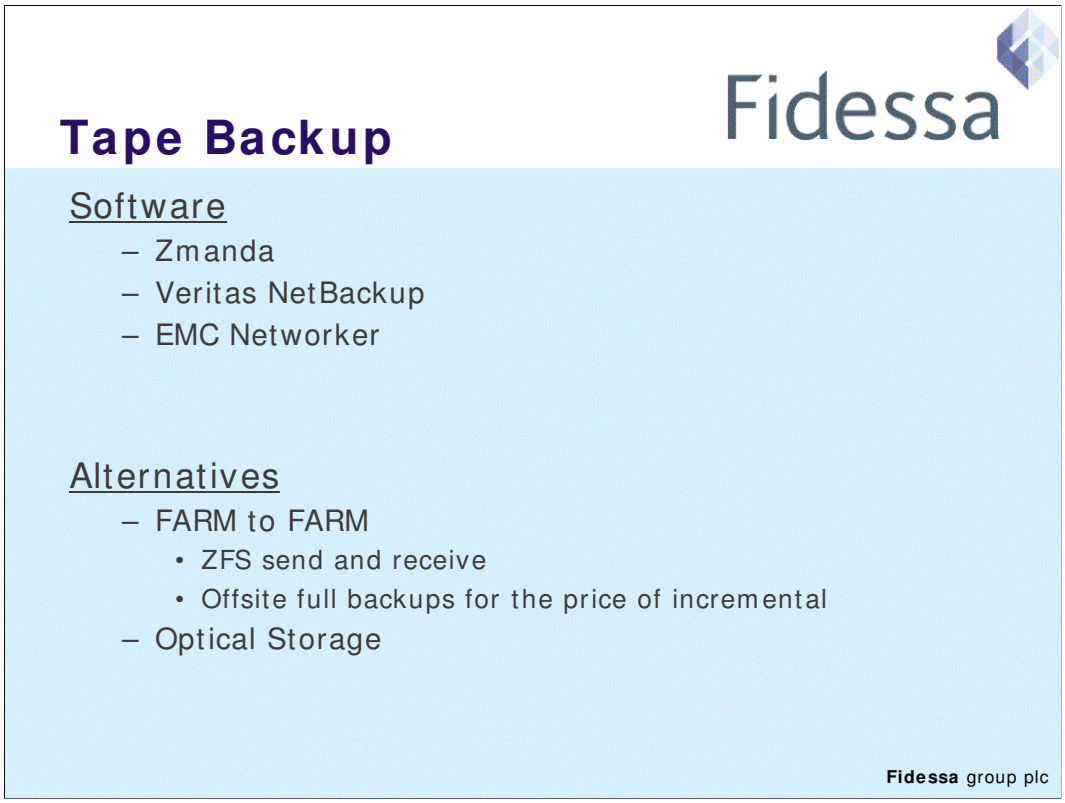
In fact, with ZFS and our client account on the FARM, then we can allow our application support staff to wander over to the FARM itself or to access their filesystems via NFS or CIFS where they can look at all the historical files themselves – no backup client or restore request required. The whole ZFS snapshot solution makes it so easy for them as all the files are "available" in each snapshot directory, they won't necessarily have to wander around multiple snapshots to find when they last changed a file. For a full filesystem restore, going to UFS the normal tools of gnu tar / cpio / dd can be used and for ZFS, you just send and receive again.

The next main point with the FARM are the tape backups.

Numbers:

Initial: Max 114GB, Avg 8.5GB,

Daily: Max 6.9GB, Avg 0.68GB



Writing data to tape with this solution doesn't really change from standard enterprise solutions. The main difference being that you are backing up a single server in each FARM, with clients being limited to filesystems on the backup server.

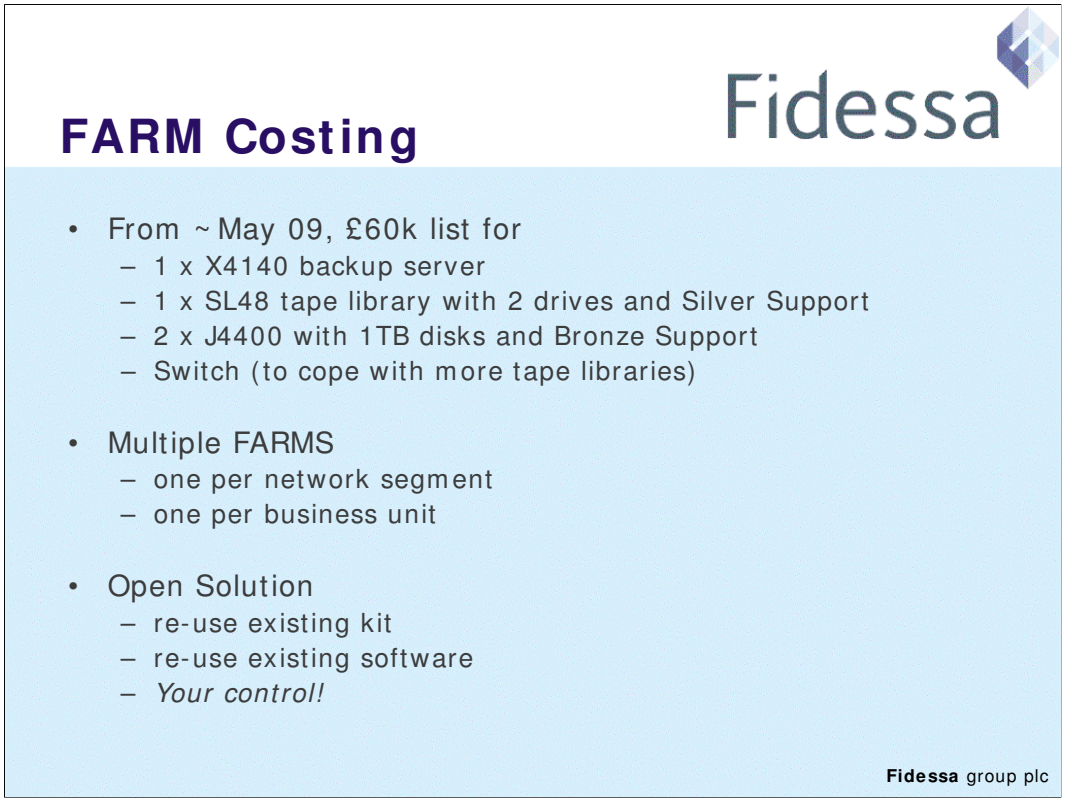
If you've got software that you're already familiar with and licensed for, there's no reason not to use it. The testing that we've done used Zmanda due to its links to OpenSolaris and ZFS support. Also, it's quite cheap – especially considering that you only need one backup server license.

The main benefit with this solution is that due to ZFS, we can backup full dumps without having to either piece together incrementals or have full dumps over the network. Also, we can backup to tape all business day without any impact to our production systems. Our testing showed that we should be able to backup our 240 clients with only 3-4 hours of actual tape time.

Other alternatives to tape backup would be to copy data from the local FARM to a remote FARM. Here we would see the full benefits of ZFS send and receive straight away, rather than the extra time for rsync incrementals. Again, each day would still be an incremental backup and you would achieve full off-site backups for the price of incremental backups over the network.

And when optical storage really kicks off, it could easily be integrated into such an open solution as this one.

So, how much is a FARM?



The numbers here were gathered in early May and are for the hardware that we used in our Proof of Concept.

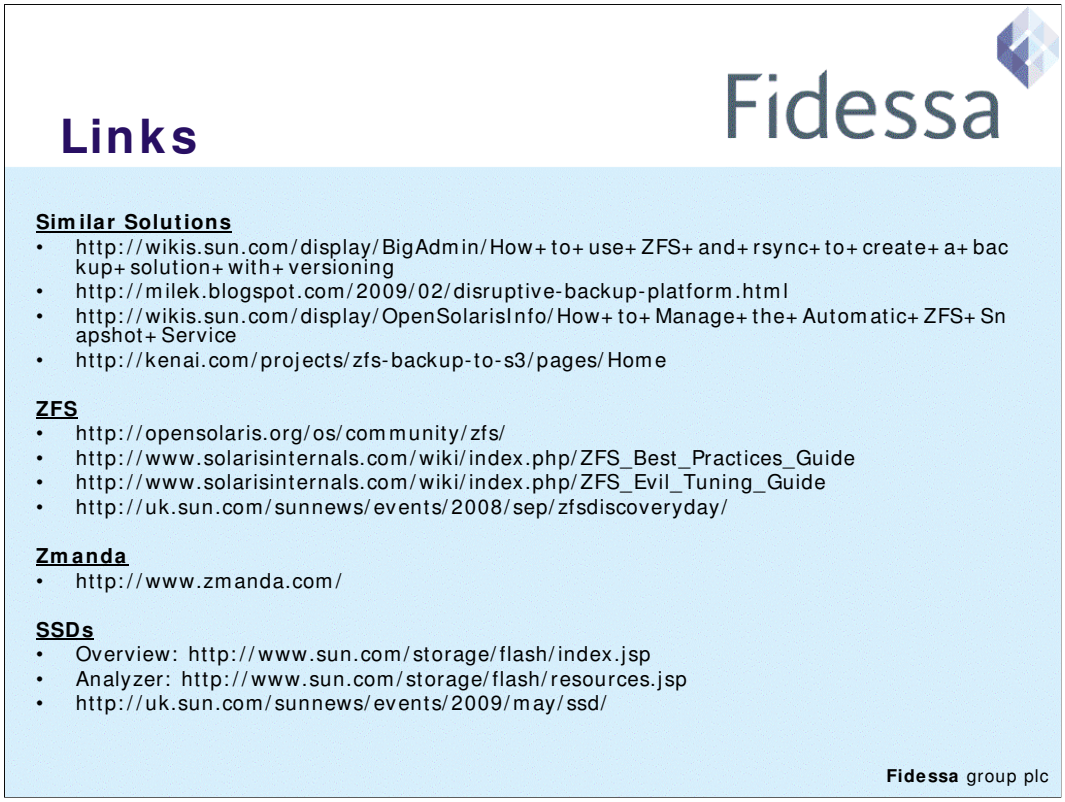
The list price of our FARM – with 2 of the disk arrays - is about £60k without VAT.

Of course, the real benefit here is that you can use what you've already got. There is no restriction to using this particular hardware. If you've got a surplus of old photons around – hook them all up…ZFS loves dumb disk.

Don't do the tape library if you don't need to – maybe you only really need to keep data for a couple of weeks, this way you can.

If you've got lots of servers, maybe you want to break it down…one FARM per network segment, one FARM per business unit. It's all flexible.

Really, the bottom line is that it's all down to you - it's all under your control!



So I'll leave you with these links and open the floor to questions – those that haven't already been asked, that is ☺