Building a low cost, highly scalable, enterprise level computer forensics lab

Foreword

Desktop hard drives are becoming bigger and bigger. However, server hard disks are not growing up at the same rate.

At present, you can buy a 1.5 Tb Sata hard drive for about 150 €.
A common 146 Gb 2.5 SAS hard drive costs about 250 €. The Cost to Megabyte ratio means SAS technology is 15 times more expensive.
This is a problem. Desktops are getting bigger but you need even bigger storage to store all data related to an investigation and it must be kept also online and secure. Currently, bigger hard drives are affordable but very fragile. Broken hard drives are quite common, much more than in previous years.
So... you need:

1. The **flexibility** from high-end storage solutions, like SAN to allow the storage to grow with your business
2. The **cost-effective** of actual desktop hard drives, trying to lower the cost-per-megabyte
3. **Availability.** No downtime, ever
4. **Redundancy.** Your data must resist both hard drive and server crashes

In addition, it should be a very good idea to reach also these goals:

1. **Security:** judicial data is very sensitive. The data should be available only to the technicians/investigators who must work on that case, even if they have administration rights on their analysis machine
2. **Multi platform support:** Computer forensics is intrinsically multiplatform. You need to use Unix, Windows, Mac OS X or whatever. So it's good to use a common network file system

Technology

We began our business with file servers with NFS. Simply too many problems. No security at all, network addressing tied to server name and share name. NFS was only fast. SMB was not faster than NFS, and was only better when it came to security. When the server number grew we had trouble finding data around the local network.

Building a SAN gave us only new headaches: we needed new expensive hardware for the infrastructure (Fiber Channel controller and switch), and both disks and storage units were much more expensive.
Otherwise we still needed some way to share the SAN file system to the analysis machines.
We looked for something new in the market. We found many cluster file system like ocfs (http://oss.oracle.com/projects/ocfs2/), lustre (http://wiki.lustre.org/index.php?title=Main_Page) and others, but they were too tied to cluster logic.
We found some interesting network file systems like CODA (http://www.coda.cs.cmu.edu/) but it lacks multi platform support and its development is incredibly slow.

At the end we found OpenAFS (http://www.openafs.org). OpenAFS is a very interesting technology:

1. Stable: it's been around since 1989
2. Multi platform: many Unix flavors, Microsoft Windows, Apple Macintosh...
3. Secure: kerberos-based Authentication, encryption, strong ACL
4. Enterprise Level: Used by many Fortune 500 companies, Universities, Research Centers, government organizations
5. Scalable: You can start with a single computer to reach hundred machines in a single cell
6. Perfect use of the bandwidth
7. It's open source! It's open and it's free.

We started building the first OpenAFS Server on December 2007 and we finished on the first days of January, 2008.

The first tests were very promising, so we began to migrate our infrastructure to Open AFS.
Now, one year later, we have:

- Three different forensics labs, in three geographic locations
- Two kerberos domains and two different AFS Cells, one for development, one for production
- 40 Tbs storage capacity
- 10 different servers

This is the whole story...

Forensics lab building blocks

Three tier client-server

The main ideas behind the lab are:

- **Reduce data moving:** Forensics labs use huge files. Every time you move data around the net you loose so much time.
- **Massive use of application servers:** if you don't want to move data around you have to move applications and processes from the client side to the server side. Your server must change from simple file servers to application server. This idea perfectly fits with the grid architecture of OpenAFS.
- **Homogeneous on server side, heterogeneous on client side:** Our storage/application servers are all based on OpenSuse 11. Analysis workstations are Windows, Unix and Mac OS X machines, both real and virtual.

So we have three different layers:

1. **acquisition computers:** these are the machines we use in the field.
2. **analysis machines:** desktop workstation to work on data, refine researches, write reports, work on mobile phones and so on.
3. **application/storage server:** a grid of commonly assembled PCs joined together in a single storage unit. These are used both to store data and to perform massive searches within huge amounts of data (a sort of e-discovery)

Acquisition computers

These powerful workstations are used on the field. When you have to acquire data, following LEAs, copy machines in a customer's location you need these.

These computers are built around a middle tower cabinet with a strong handle to carry the computer around. Notebooks are good only when you need to fly, otherwise are too limited in I/O speed and storage capacity.
Inside a Supermicro X8SAX mainboard we have: (http://www.supermicro.com/products/motherboard/Xeon3000/X58/X8SAX.cfm) 6 SATA 1 Tb hard drives, 1 SAS controller, 1 Quad-Gigabit ethernet, and a SCSI Ultra 320 controller. Operating system is GNU/Linux and all SATA drivers are joined together as a single Software RAID level 6. We prefer software RAID, instead of an hardware one, because it's more usable and it's simple to recover if something fails.

The quad giga ethernet can be used as 4 different interfaces (useful, for example, to acquire 4 different machines full band once) or joined together into a single trunk to speed up connection (with a little help from the main switch to create the trunk).

Computer Forensics Live distributions, as DEFT or CAINE are good companions for our technicians and helps them to boot target machines with a safe OS to clone them through the net.
The policy of our labs is to keep these computers clean. They should always be ready to gather data. So when we come back from the field these computers are immediately connected to the LAN to copy all data in the back-end. When the copy ends, the data raid (usually mounted on /mnt/repository) is cleaned with a “cat /dev/zero > /dev/md0”).

Analysis Machines

We used three different kinds of analysis machines.

At the hardware level they are quite similar to acquisition computers, they have only less storage since they are connected directly with the common network file system.
Investigators have full power on the local workstation so they can launch command as “root” via “sudo” on Unix and have also “Administration” account on Windows computers.
Storage back-end can be reached in two different ways:

1. **On Unix machine** pam (Pluggable Authentication Module) are changed so investigators can login only using their kerberos account. Home directory are in the OpenAFS so the environment is the same regardless of the machine used to login. In particular, we use saslauthd daemon to let the system searching in the LDAP database, and pam_krb5 to authenticate the user defined inside the LDAP database toward kerberos and to retrieve token for OpenAFS in order to mount the home directory at the end of the logon process.
2. **On Windows computer** people use an Administrator account to login locally and then MIT kerberos Client and OpenAFS modules to access network file system. We prefer this solution instead of joining the kerberos realm directly since it works flawlessly also with Home version of Windows that we often find right installed on portable computers.

Investigators are able to work on their computers performing every kind of operation (both user level and root level) but they are able to see only the cases assigned to them. There is only one way to retrieve case data: logging through kerberos on OpenAFS. Even if you are “root” on a OpenAFS server you can't access the data on it.

Unix machines are common X86_64 computers with a

OpenSuse 11. Every installation is performed with a standard graphical system with “development” and “kernel development” profiles added.
We install “screen” and “free-nx” on every analysis machine, so it's possible to launch even very long process, safely connecting (using ssh) through the net. When you have to leave you simply detach the character “screen” or the graphical session (via free-nx) from your notebook and the process will continue its work without any problem.
All forensics software is installed on a specific volume in the OpenAFS. Adding a new computer it’s not an issue. Install a new computer, adding OpenAFS client and everything, from home directories to every software is just ready-to-go.
In our labs we are using, under Unix (just some examples):

- **The Sleuth Kit** (http://www.sleuthkit.org/): fundamental for everybody who wants to play with Open Source CF
- **Pyflag** (http://www.pyflag.net/cgi-bin/moin.cgi): Far better than Autopsy
- **MySQL** (http://www.mysql.org): Pyflag works on MySQL. We have a specific DB Server just for it.
- **Foremost** (http://foremost.sourceforge.net/): file carver made easy
- **dhash** (http://www.deftlinux.net/wiki/index.php/Dhash): fast multiple hash calculating program
- **dcfldd** (http://dcfldd.sourceforge.net/): the standard open source forensics variation of “dd”
- **wireshark** (http://www.wireshark.org/): useful in network forensics
- **Xplico** (http://www.xplico.org/): network dissector for tcp application layer

Open Source Computer Forensics tools could seem quite raw compared to some new “bells and whistles” commercial computer forensics packages, though not only being the most up to date, and also the only available way to investigate some “not so widespread” computer architectures and information systems.

Windows analysis machines have around three useful main tools we experienced along the last years.
The first one is **X-Ways Forensics** (http://www.x-ways.net). It's very good computer forensics environment.

We tried many different computer forensics software: Encase from Guidance, UTK/FTK from Access Data, P2 Commander from Paraben, X-Ways Forensics from X-Ways.

The last one is an excellent product, showing some advantages over those other competitors:

- It's small: The core is about 3 Mbyte, the entire program (viewer included) is no more than 15 Mb. It can be copied in a USB Thumb drive and launched from it.

Has a very good core: X-Ways Forensics is built around WinHex, which is a killer application among hex editors in the Windows world.
Has many interesting functions, an internal database engine lighter than Oracle, the Stellant Software viewer and it has a very fast development cycle.
Side-by-side we have the software we use for mobile forensics analysis. It's obviously based (no serious open source software is available at the present) on a Windows machine and two different mobile forensics products, Oxygen Forensics Suite 2 and UFED Cellbrite. We choose these two products because they have a very large support of European GSM phones, a strong smartphone support (Windows, Mac OSX and symbian ones), and a very good technical support. UFED is also a good standalone product for on the field operations.

All of our Windows machines use nVidia 8800 GT and are joined together as single cluster with Elcomsoft distributed password cracker with GPU support. We can speed up to 1.500.000.000 ntlm passwords per second in password cracking.

The Back-end

We are proud to say that the back-end is the gem of our lab. It has the all the features of an high-end solution with no disadvantages discovered until now, and it costs a fraction of the price. It's based upon two main components: an authentication server and a cell of OpenAFS servers.

Authentication Server

We decided to use an Unix machine with a MIT kerberos V Server for authentication and OpenLDAP for accounting.

We chose kerberos because it's a standard protocol, it's supported by many different services (included OpenAFS as common network storage) and operating systems (Unix, Windows and Mac OS X work all well), it's very secure and useful to implement a single sing-on solution among all forensics lab.
We have a master kerberos and LDAP server, aided by a slave kerberos and LDAP server in every remote site.
The only change we applied to the default configuration has been to increase the lifetime of the ticket from 10 hours to 3 days (many computer forensics processes run just too long).

Application Server and OpenAFS

OpenAFS is a wonderful piece of software. Compared with other network operating system and protocols it is

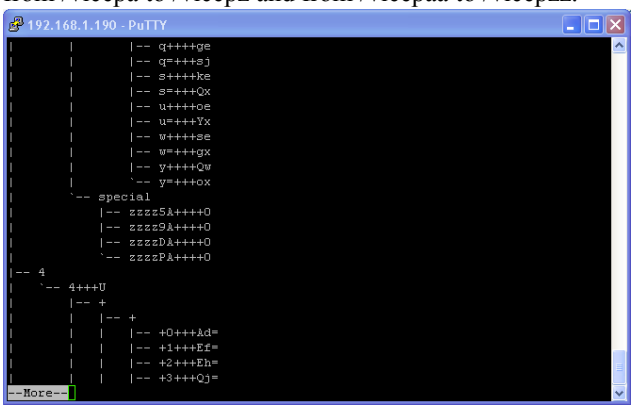
simply ahead.

First of all you can forget about the basic concept of a File Server. Many operating systems force you to use the server name or address in the network path to reach the shared folder. NFS and SMB are two classical examples.
This is quite frustrating if you have many file servers in your network. Your data is splitted through many different computers and it's just too hard to remember where the data is.
To avoid this problem you could think to shrink all of your server in a “big iron” with a SAN behind.
This move should help you to minimize the fragmentation problem but you have to spend much more money to use enterprise level hardware and, moreover, you have to say “bye bye” to the computation power of your distributed network. If you plan to use it on file server service this is not a problem but, in a computer forensics environment with many enormous hard disk image files, it's not a good idea. Every time you move one of this files through the net, even full gigabit ones, you slow down the entire analysis process. It's far better to use your file server also as an application server, moving all data intensive processing server side. If you have a single file server you can't work on many cases at the same time.
OpenAFS can help you obtaining a common storage shared among many servers.
First of all, you must think you have a single tree in every cell (which is a set of file servers). This network tree is comprised of “volumes”.
Note: When you log into the cell you are authenticated automatically toward all file servers in the cell. Also system administration need to have just an administrative account in the cell and can be performed from every machine in the cell, a client or a server as well.
Volumes are some sort of “piece of file system”.
They can contain files, directories and mount points for other volumes, so they can be nested within each other. The base volume of the tree is always the “root.afs” volume (mounted on /afs), followed by “root.cell” mounted on the name of the cell (for example forensics-lab.com) inside /afs (ex. /afs/forensic-slab.com)
Inside the root.cell volume there will be the mount points to the other volumes distributed among all servers of the cell.
Two other interesting concepts about OpenAFS.
The first one that **OpenAFS uses a Unified Named Space**, for every cell. It's always the same regardless the operating system in the client or server side. So every machine in the cell has the same view of the network file system. Even in Windows you only need to map a single network drive where you see the entire afs tree.
The second one is about volumes and how the client side finds the location of the volume inside the cell. There are two main components inside OpenAFS, one client side (**Cache Manager**) and one server side (**Volume Locator**).
OpenAFS uses asynchronous logic. Clients are never directly connected with the file servers. Instead the Cache Manager asks the Volume Locator server where the file is. Then the correct file server sends the file (if it's small) or a part of it (if the file is large; useful when you are using image files) to the cache manager. Then the client will use the data present in its cache. When the user chooses to save, the cache manager will send the new file (or portion of it) back to the server. If someone else changes the file on the server in the meantime, the Volume Locator will advise all other client to refresh their cache as well.
Note: This works incredibly well also in WAN environments. The INFN (Italian National Nuclear Physics Labs), for example, has a single cell with various sites in many areas of Italy.
The system works incredibly well. Tuning client cache to use 128 to 256 Mbyte of memory cache, you can mount a dd image remotely through loop devices as if it were a local file. You can send a dd image to the remote storage (using a Gigabit ethernet) at 60 Mbyte per second as sustained transfer rate.
This architecture gives OpenAFS many other features. First of all you can do everything you want with the volume. **You can move, replicate, take a snapshot, dump or backup one of them anytime, without a single downtime and when the users are working with the volumes.**
As a consequence you can add as many servers as you want in the cell. The idea behind OpenAFS is quite similar to grid computing, if you want more storage you only have to install a new server. The system is scalable and flexible enough to grow up along with your needs. The process is simple. Install the operating system, add the server to the cell with two “bos” commands and it's done.
Then you can move volumes directly to the new server, without any impact to the network tree.
Our typical OpenAFS server is an assembled computer like this:

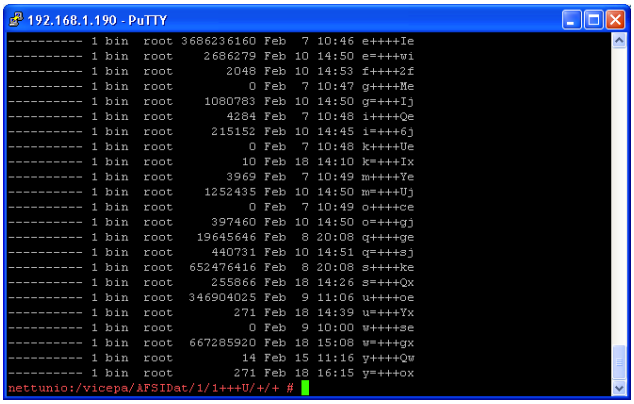
Cabinet:	Cooler Master G-lite
Mainboard:	MSI KA9A2 Platinum
CPU:	AMD Phenom 9500
Memory:	4 Gb
Graphic card:	8800 GT
HD:	6 x 1 Tb SATA HD for a single 5 Tb software RAID 5.

This server cost us no more than 1300 €. If you try to calculate cost per gigabyte is about 0.21 Eurocent.
It should be about 1.7 € per Gigabyte if you plan to use SAS disks, considering just the cost of the disks. No cabinet/rack, nor fiber channel hub has been considered. So the total price should be even higher.
If you plan your OpenAFS wisely, you can use cheap hardware. You can swap disks if they break (software raids compensate the problem). If you carefully plan replication of the volumes and backup (both are native functions of the OpenAFS and you don't need anything else to implement them) you can also loose an entire file server without loosing data. So file server can be

thought just as simple building blocks of the entire network file system.
They are just a brick and can be swapped easily.
Another interesting feature of OpenAFS is how it can store data on its server. OpenAFS wants to save his data on **entire file systems** (i.e. they can't be a piece of the root filesystem of the machine) mounted on directory from /vicepa to /vicepz and from /vicepaa to /vicepzz.



Picture 1: An example of the complex tree of an OpenAFS



Picture 2: Files in the raw OpenAFS Filesystem

OpenAFS uses these file systems as raw devices. It saves data in a very strange way, splitting everything in a incredibly complex tree of hundreds of directories.

It's nearby impossible to rebuild data without the help of OpenAFS daemon.
This means that, **even on the server**, you can't access to the data without logging in the kerberos and getting the token. So even the server can access data only acting as a normal client for the OpenAFS net.
When you work on OpenAFS as client you are tied to the strong security of the file system. OpenAFS support multiple ACL per directory defined per user or group.
OpenAFS supports 7 different permissions:

"l" lookup, permission to view the contents of a directory
"i" insert, insert new file in a directory
"d" delete
"a" administer, the permission to change ACLs
"r" read
"w" write or modify files
"k" lock

In our policy, forensics technicians have all permission apart "a" and "d", so they can write and modify but cannot delete anything and changing ACLs.
If you **carefully plan the ACLs, configure "sudo" in a wise manner and log everything in a remote log server**, you can use servers as additional analysis machines. The main advantage is that servers can access their own volumes through a "network loop device" which is much faster than the real net.
So we installed all the OpenAFS server with the same software of the analysis machine. From our point of view they are essentially the same. Simply, server have mount points to raw OpenAFS data.
This approach helped us, during the last year, to save probably weeks of work moving server side the most I/O intensive operations. Think about keyword searches, distributed among many server, every one of them searching on its own volumes.
Another function which seems to be created just for computer forensics is snapshot. You can freeze the state of a volume in a certain time. This is an incredibly useful feature to avoid to change the state of the evidence even if you plan to perform some test that inevitably modifies the evidence. You don't need to copy huge files many times. Simply create a new snapshot every time you need.
The only problem we have is with database files. Since we are using pyflag (which uses MySQL as back-end) we created a separate MySQL server since even the Administration manual itself doesn't advice about how to install a kind of RDBMS into an OpenAFS volume.

Conclusions

This labs grow up with us in the last two years. Working both with Open Source and Commercial Computer Forensics tools helped us to choose the right tool in every case even if the target computer wasn't a classical Windows machine.

On the other hand working with OpenAFS helped us to investing our budget on learning (and training to the new ones) computer forensics instead of buying enterprise-level hardware and software and adding computation and storage power a brick at time. We learnt that the difference is done by the humans skills not by the cost of the technology used.