Information Assurance In A Distributed Forensic Cluster

By

Nicholas Pringle and Mikhaila Burgess

From the proceedings of

The Digital Forensic Research Conference

DFRWS 2014 EU

Amsterdam, NL (May 7th - 9th)

Digital Investigation 11 (2014) S36–S44

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

Information assurance in a distributed forensic cluster

Nick Pringle*, Mikhaila Burgess

University of South Wales (formerly University of Glamorgan), Treforest CF37 1DL, UK

Keywords:
Digital forensics
Distributed processing
Media analysis
FUSE file-systems
Information assurance

A B S T R A C T

When digital forensics started in the mid-1980s most of the software used for analysis came from writing and debugging software. Amongst these tools was the UNIX utility 'dd' which was used to create an image of an entire storage device. In the next decade the practice of creating and using 'an image' became established as a fundamental base of what we call 'sound forensic practice'. By virtue of its structure, every file within the media was an integrated part of the image and so we were assured that it was wholesome representation of the digital crime scene. In an age of terabyte media 'the image' is becoming increasingly cumbersome to process, simply because of its size. One solution to this lies in the use of distributed systems. However, the data assurance inherent in a single media image file is lost when data is stored in separate files distributed across a system. In this paper we assess current assurance practices and provide some solutions to the need to have assurance within a distributed system.

Introduction

The notion of using distributed processing to address the increasing scale of forensic investigations was first considered in "Breaking the performance Wall" in 2004 (Roussev and Richard, 2004). Despite being revisited several times since, (Ayers, 2009; Beebe, 2009; Garfinkel, 2010; Pringle and Sutherland, 2008; Richard and Roussev, 2006; Richard et al., 2007), this has not been developed and adopted as a workable solution. There has been a resistance to the idea of using an architecture where the data is moved and stored on a multitude of hosts for processing. In this paper we briefly consider the technical issues but conclude that the most important reason is the lack of a forensically sound approach to ensuring information assurance within a distributed system. This is required to ensure evidence management is regulated and clearly accountable for the legal community.

We will introduce our design for a middleware distributed processing solution, FCluster, which is specifically designed to provide assurance for the integrity of data.

Background

As digital forensic investigation methodologies have matured to accommodate the developments in technology, crime and investigative capabilities over the last 20 years, internal controls have been introduced to provide assurance standards required by the legal process.

Within our expectations of assurance there are a relatively small set of acceptable and 'trusted' investigative tools. FTK and EnCase are two of the most popular and trusted tools for digital media forensics. We know from more than a decade of use that their design endows confidence in the investigative process, and this is supported by these tools being tested for forensic appropriateness by NIST. In particular, the risk of 'mixing up data' between the evidence media and the host computer is negligible. There is no realistic way that data from another image could be introduced because there is no mechanism, other than operator error working on the wrong image, for this to

* Corresponding author. Tel.: +44 0 1443 4 83261.
E-mail address: nicholas.pringle@southwales.ac.uk (N. Pringle).

http://dx.doi.org/10.1016/j.diin.2014.03.005

happen. Provided the investigator is trained to use these applications as they were intended, the system is inherently assured. The designers consciously choose not to have a write-ability, not because it's just easier that way but because we have a special need to protect the data under investigation.

Assurance standards applicable to digital forensics

Unfortunately there are no explicit rules to define Information Assurance for processing Forensic data. Forensic evidence must adhere to the Daubert principle and the Federal Rules of Evidence in the US, ACPO guidelines in the UK (ACPO, 2012) and corresponding criteria elsewhere. ISO 27037 (ISO 27037:2012, 2012) addresses the acquisition and preservation of digital evidence but uses language such as "protected as far as possible" and that "evidence should be stored in an evidence facility that applies physical security controls". Standards like ISO17025:2005, intended for 'chemical' laboratories, have been the basis of digital forensic facilities but the translation from the analogue to the digital world is not always easy. ISO 27001:2013 defines characteristics of a management system that provides assurance, but not assurance itself. PCI-DSS (PCI Security Standards Council) does provide a more prescriptive standard but doesn't map well to digital forensics. When these are appropriate, unfortunately they are generally based upon the vague notion of 'best practice' and 'the accepted norm' in the particular field. It is difficult to apply in a rapidly developing domain, such as digital forensics, as technology changes are naturally always ahead of 'best practice' developments.

Internal controls in digital forensics

In practical terms, these reveal themselves in some of the characteristics of an existing system when, for example, a new item of evidence is introduced into the lab. It would first be recorded in some form of log. When the evidence image is copied onto the storage facility its success or failure needs to be validated, perhaps with a cryptographic hash digest, for example SHA-1, and this is recorded in the log book. The hash digest is an inherent property of the image. If the validation fails, the operator would investigate the process or equipment and make remedies and rerun the copy. This time, hopefully, it would succeed and the task is complete. Its success, and the previous failure, should both be recorded on the log book. In a paper system, the log book should have certain characteristics. The pages should be numbered and bound together. Anything written should be in ink. Lines on the page should either have writing or be lined through. If the log book is implemented on a computer system there should be an external verification, for example a time date stamp encrypted by PKI, that is beyond the capabilities of the operator to amend. These sorts of controls are common and should be familiar to any investigator.

All these processes should be subject to an *Audit*. By *Auditing*, we are checking that the system worked. The main problem with Auditing is that it is reflective and it often implies a protracted period of time passing before the audit. External audits are often annual, internal audits are perhaps, quarterly. It addresses issues that occurred in the past, assesses their conformance or non-conformance and should trigger changes in the system to prevent further breaches.

This was the case in the quality control employed in most industries in the Western World after the Second World War. Generally, goods were manufactured and were subject to quality control as a final stage where a sample set was tested for conformance. Those non-conforming were removed and either reworked or scrapped. The audit would trigger a period of reflection and perhaps modification to the production system to reduce the failure rate. Regrettably, there was an acceptance that a percentage of non-conformances would get through the system.

From audit to assurance

During the 1960s the Japanese introduced the idea of total quality assurance. The most important aspect of this was that controls were introduced before that action took place, not after.

The dictionary definitions give a sense of the retrospective nature of an audit (Dictionary.com, 2014) and the future intent of Assurance

Audit (*noun*)

1. **an official examination and verification of accounts and records, especially of financial accounts.**
2. **a report or statement reflecting an audit; a final statement of account.**

Assurance (*noun*)

1 **a positive declaration intended to give confidence; a promise**.

synonyms: word of honour, word, guarantee, promise, pledge, vow, avowal, oath, bond, affirmation, undertaking, commitment

2 **confidence or certainty in one's own abilities.**

synonyms: self-confidence, confidence, self-assurance, belief in oneself, faith in oneself, positiveness, assertiveness, self-possession, self-reliance, nerve, poise, aplomb, presence of mind, phlegm, level-headedness, cool-headedness

Japanese production lines did not produce faulty goods because faulty components were not allowed to enter the production line. The effect of this change on the industrial base of the western world is a matter of history. During the 1970s and 1980s products from Japan surged leaving their North American and European competition behind, being viewed as unreliable. Modern management systems like Total Quality Management and Six-Sigma have their focus on controlling inputs and processes during the manufacturing process. Increases in quality, and customer satisfaction, are natural consequences of this approach.

Assurance in current computer systems

Most digital evidence from storage media presented in court is the result of analysis conducted using FTK or EnCase. This is so much a de-facto standard that we rarely question it but both systems are based on the same principles and on more than a decade of acceptance and precedence. At its heart is the idea of always presenting evidence originating 'from the image'.

Imaging tools usually make a cryptographic hash digest of either sections of data or the whole media. When the investigator copies the image onto the laboratory storage facility *they should* run a program to create a new cryptographic hash digest and compare it to the original to confirm the data is unchanged. There are a number of imaging programs used with varying assurance. The dd utility has no internal check-summing facility; both Expert Witness Format (EnCase) and Smart use file structures within their images to checksum every block, typically 64 KB. We are assured of the integrity of the data because it is seen as one complete, wholesome entity and is internally consistent.

It is largely left to the administrative system built around the computer system, as outlined in section 4, to provide assurance with existing tools that store or process these images.

Concepts like "Chain of Evidence" or Provenance have existed in legal proceedings for some time. Although users will take care of their data, the legal profession does pride itself on its highest possible standards in this matter. The ACPO guidelines describe this as a key task of the forensic practitioner.

Distribution of both data and processing

Current systems largely assume that the investigator will be handling a relatively small number of media items. In many investigations this might be only one or two forensic images. This is changing because case volumes are increasing (Justice FBI, 2012). To cope with this, it has been suggested that the next generation of forensic software could adopt a distributed processing model.

At this point we should make a distinction between distributed processing with centralised storage and distributed processing working with distributed storage. Having a distributed processing architecture that relies on a central, non-distributed, store of forensic images (Fig. 1) implies that the data has to be distributed to the processing nodes before it can be subjected to processing.

This is the case with FTK's 'distributed' processing. Processing time with this topology is dependent on the connection between the switch and the file server which rapidly becomes overloaded and limits scalability. We can mitigate this to some degree by building a storage facility based on fast SSD storage (450 MB/s), SATA III (600 MB/s) interfaces and even 10 Gb (1000 MB/s) Ethernet networking but this can be prohibitively expensive. Even this has limited capabilities in scaling out to even tens of processing hosts. Assuming we can make this investment it can still take many hours just to read the image off the storage media. If we wanted to conduct simultaneous analysis of several images held on the same storage facility it would have a significant impact on data dispersal time and so overall processing time.

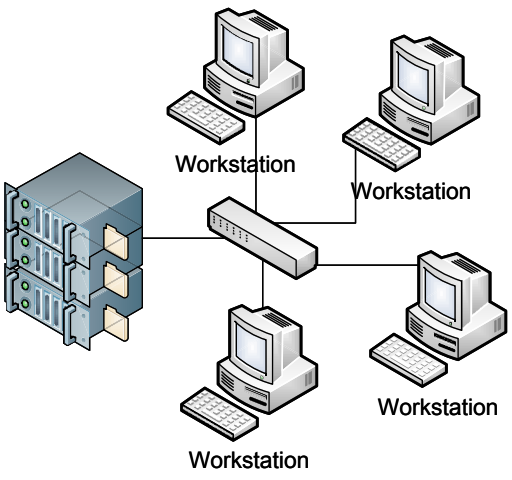

Fig. 1. The most common current architecture.

Digital forensics is not the only domain that has encountered this type of problem. Recently, Google solved their huge data problem by developing and applying a truly distributed data storage and processing model called Hadoop/MapReduce (Dean and Ghemawat, 2004). Although Hadoop/MapReduce provides distributed storage and processing it lacks the levels of assurance we require in processing data for presentation as evidence in legal proceedings. Distributed systems like Hadoop, Condor (Thain et al., 2005), Nimrod (Abramson et al., 1997), Weka (2014) and Globus (2014) have been slow to incorporate information assurance. Most have some access control but the users are more interested in getting their data processed than the nature of the environment in which it is stored. This is changing. Hadoop, for example, has been extended by commercial enterprises like Cloudera (2014), who realise that commercial acceptance now requires security but these are designed and built with general commercial markets in mind. In these systems it is often up to the user to exhibit diligence in the processing of a job. It is quite acceptable to not know where a file is stored or where it is processed; in fact it's a feature of Cloud Computing. There is some audit trail but rather like event manager in Windows, it is intended for performance and debugging issues rather than assurance. Information Assurance has greatly improved within Hadoop since Cloudera released CDH3 in April 2011 but it is unlikely that it will ever be extended to the exacting requirements of the legal process.

In all of these cases, information assurance has been added on as an after-thought. The assurance inherent in the legally established idiom of one investigator, one machine, one image would not be upheld in any current distributed storage/processing system. It's entirely understandable why software vendors don't support true distribution. It's doubtful that anybody would buy the product.

It would be much better if we could adopt a truly distributed storage and processing approach but built on a foundation of an assurance system rather than amend the existing systems.

Images, digital evidence bags and SIPs

The practice of acquiring a digital crime scene in the form of a 'forensic' image has served us well over the last 20 years but it is now under considerable pressure because of the size of media needing to be imaged. We feel this will lead to an increasing adoption of smaller units of storage. Well known formats such as AFF (AFFLIB) and DEB (Turner,

2006) have been established. We will refer to these collectively using a term taken from the ISO 14721, Open Archival Information System (OAIS) (OAIS, 2014), Submission Information Package (SIP).

Assurance in distributed storage

As soon as we split 'the image' into the SIPs that are needed to enable distributed processing we lose most of the inherent integrity of the 'oneness' of the image. Suddenly we have, perhaps, hundreds of thousands of SIPs to validate. In what way could assurance be re-established?

Information Assurance while processing SIPs could be provided in practice using some of the following methods:

- By a Property of an object: making and testing Checksums, Check digits, size, Control totals.
- By the Position/Location of the object: the fact that a file is in a certain location further enhances our faith that it is the correct one.
- By Loops of Authority and Acknowledgement: only accepting data from a device that was authorised to provide it.
- By Access control: allowing and denying.
- By Separation of process: having functionality provided by more than one program and clearly separating stages by function.
- By Audit trail: requiring independent sequential stamp, indelible records, recording with an authority.
- By Checklist: testing to see if previous checks have been completed and recording them in a table.

Introducing FCluster

FCluster is a middleware that provides an environment to allow forensic data processing to proceed with assurance. It is a means by which data integrity can be controlled; it is not an application program.

The design is based on the following assumptions, derived from current practice and technological developments.

- Media will continue to grow in capacity and quantity.
- That 'cross drive' forensics will be of increasing importance as individuals have many storage devices, crime is becoming more organised and forensic analysis systems will increasingly have to address multi agency interests.
- That multi-agency investigation will increase but will experience problems sharing and transferring large datasets.
- That, because of the above, the notion of the 'image' is becoming untenable but will be required for some time as a legacy. Instead evidential data will need to be stored as separate files across the system. Consequently we must expect tens, if not hundreds of millions of files in a system that stores the contents of many forensic images.
- That in most investigations, the evidence is found in 'the obvious place' and that most investigations are a case of locating and recording data found.
- The system should allow existing legacy software to run where possible. This new system should not require Guru programming skills with knowledge of devices like GPUs to gain access to huge processing power. However if such programs are developed it should allow these as well.
- That quantity of data requires the development of more automated tools. These can be simple reporting or correlating tools but need to run against large datasets.

The FCluster architecture

FCluster is a peer-to-peer middleware for a network of heterogeneous host computers. The prototype is built on Ubuntu Linux with future development planned for Windows and MacOS. Most of the code is either in C or Bash scripts. It uses MySQL, libcurl, ftp servers and ntfs-3g.

FCluster SIPs

FCluster includes a design for an SIP with a simple structure which only works with NTFS file-systems. This was done for the sake of simplicity when developing the prototype.

An FCluster SIP comprises of 2 parts (Fig. 2). An extensive header section contains XML delimited meta-data about the file's place on the original evidence media. This includes data from the file's entry in the NTFS $MFT and also a list of cluster numbers the file originally occupied on the source file-system together with an SHA1 for each of the clusters. The data section holds the file data which is encrypted using AES-256, with the key sent from FCluster, and then UUencoded to reduce problems in portability.

The SIPs themselves are named in a regular manner, [*VolumeID*]-[*SHA1*].meta. When the SIP is finally unpacked, decoded and decrypted on the FCluster the resulting file must have the same SHA1 as its filename suggests and is included within the header section of the SIP. To achieve this it must have been generated on the imaging device authorised by the key created by the FCluster when it authorizes imaging (see section 14.1) or it will not decrypt when it is ingested into the FCluster file system. These form two assurances, one of a property of the file, the name and the 'double entry' of the success of the encryption/decryption key.

FCluster subsystems

Applying a principle of separating processes, FCluster comprises of 4 sub-systems (Fig. 3).

These are explained in more detail in section 14.

<metadata ntfs>
This contains detailed information about the directory entry and details of which clusters the file occupied, together with SHA1 checksums for each cluster
</metadata>
<data>
This contains UUencoded version of the data that was encrypted with AES-256 with the acquisition authority key
</data>

Fig. 2. FCluster SIP structure.

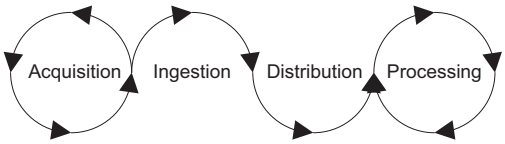

Fig. 3. The 4 subsystems with FCluster.

FCluster roles

Each host in the cluster might fulfil all the FCluster functions listed below, but it is likely that most hosts will be allocated just three or four roles.

- **Acquisition Authority**

That creates the cryptographic keys used to authorise imaging.

- **Imaging**

That creates the directory metadata SIPs, file data SIPs and Image files.

- **FClusterfs file-system metadata storage**

The heart of FCluster is a multi-featured File System in User Space (FUSE) file system (Filesystem using MyS, 2013) based around an SQL database.

- **SIP Ingestor**

That locates expected new evidence SIPs and triggers ingestion.

- **Load Balancer**

That chooses which storage/processing host should hold the primary copy of the data based on its workload.

- **Replicator**

That makes sure there are enough copies of the SIPs to ensure redundancy and also verification that the data is still valid.

- **Data Storage server**

That actually holds the data.

- **Processing**

Which actually does the processing. Almost always combined with the storage role.

We believe our proposed system provides assurance at every stage in such a way that the next stage cannot commence if the previous assurance is not satisfied. The core of this assurance is embodied in the use of FUSE file-system specifically designed for our purpose.

FClusterfs

We have observed that Map/Reduce implements a new file system, HDFS, as a base for its processing. This has been implemented as a middleware on top of the native file-system used by the operating system. We follow the same approach.

The use of FUSE to build custom file systems has been proposed within the digital forensics domain (Richard et al., 2007). FClusterfs advances the notion and uses the technique to provide a solution that addresses the key issues of Assurance in a distributed processing environment. It merges together several existing FUSE file systems to form a new file system.

FClusterfs is based on MySQLfs (Filesystem using MyS, 2013). MySQLfs employs an SQL database consisting of 3 tables to completely replace the native file system. The 'inodes' table provides storage for file metadata like names, dates/times, size, access rights etc usually seen as a 'directory'. The 'tree' table stores the hierarchical structure of folders and filenames found in the file-system. The 3rd table 'data_blocks' stores the actual data as a series of binary large objects (BLOBs) replacing the clusters of the disk format.

In FClusterfs we use the tree and inodes tables found in MySQLfs. FClusterfs provides read-only access and so we never need to manipulate directories. We have a table called 'meta-data' to store the meta-data from the original location of the data. This is a variable length, large text field and so is better in a table of its own.

A single FClusterfs database can store many file-systems. We have a table, VolumeInformation, which contains a record of each file-system stored within the inodes table. We have added a field 'VolumeID' to inodes to identify which file-system the entry relates to.

We substitute the functionality of the 'data_blocks' table in MySQLfs with the ability to read data stored on remote servers. We have chosen to connect to the remote servers using the ftp protocol because of the features of another existing FUSE file-system curlFTPfs (Robso, 2013). curlFTPfs allows the user to mount a connection to an ftp server and make it appear to be part of the host's file system. curlFTPfs attains much of its power and flexibility because it is based in the libcurl library and can support not only ftp but SSH, SFTP, HTTP, HTTPS but, despite known security issues with unencrypted data transfer, we have chosen ftp as a simple base for a prototype. In a real world scenario, SSH would be a more robust protocol. curlFTPfs only allows one ftp server per mounted file system. In FClusterfs we have enhanced this to be able to access individual files on any ftp server on a file by file basis. The corresponding server details are stored in the file's record in additional filds we have added to the 'inodes' table. When the user sees a directory listing in their user space it appears as a continuous list drawn from the 'inodes' table but in reality each file's data will be on a ftp server which is most likely remote. Each file is held in its entirety on the ftp server. In the prototype the entire file is transferred and held in cache in memory. In curlFTPfs 128 MB chunks are transferred just once and, if the file is over 128 MB, a mosaic is built in a cache in local memory.

It is important to realise that although FClusterfs does allow data to be transported across the Ethernet network, it

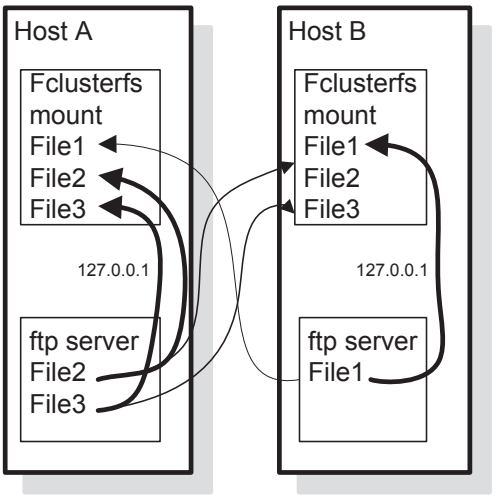

Fig. 4. ftp via 127.0.0.1 localhost loopback.

is primarily a means of standardising access to data held locally on its own ftp server. When we use the term 'passes across the network' it should be taken as via 127.0.0.1, the localhost loopback connection (Fig. 4).

FCluster is also peer to peer and so any node can mount a directory that can reference files on any server (Fig. 5).

Further, data held on this multitude of ftp servers is encrypted and uses techniques from ecryptfs (Hicks et al., 2013) to decrypt data on-the-fly. After it leaves the ftp server media, it passes across the network and is decrypted in the user's host before being held in cached space in RAM in their Virtual File System.

As previously mentioned, FClusterfs is read-only. There is no code to provide functions like write/delete/chown/chmod. This is a fundamental requirement of a forensic system and, fortuitously, greatly simplifies the code.

FCluster has auditing which it draws from Loggedfs (Flament, 2013). Loggedfs' audit is felt to be too granular for our purposes and instead we choose to record only significant actions like SIP movement, unpacking and the opening of data-files for processing. Recording access to parts of a file is felt to be unnecessary and would only slow the system and make the logs unreadable. All audit records are stored in a table 'audit' recording date/times, users.

Although the data location url information is available to the user eg ftp://myserver.com/, the username and password needed to login to the ftp server and gain access the data is not. It is held in another table 'serveraccessinfo' and is retrieved on-the-fly during a read request by FClusterfs. Users can only access evidence via the FClusterfs file-system which provides data from the ftp servers.

FClusterfs is intended to completely replace any need for network shares like NFS or SMB but to emphasise, although FCluster provides access to the file-system under investigation and will work over a network connection it is not the most effective way to work. It is intended to process local data by the host of the ftp server holding each of the files. The location, url, of the ftp server hosting the data is part of the 'inodes' table extending the fields used by FClusterfs and so the 'locality' of the file can trigger the processing task to be initiated within the host.

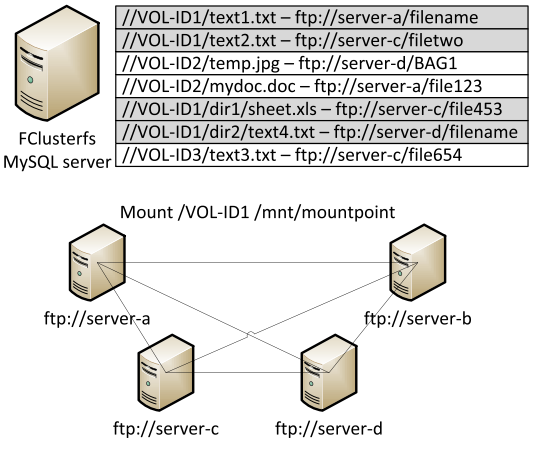

Fig. 5. FCluster mounts peer to peer.

Mounting the FCluster file-system

The behaviour of an FClusterfs file system is defined when it is mounted by a command line which contains the following entries:

```
fclusterfs
–mysql_user=me
–mysql_password=mypassword
–mysql_host=25.63.133.244
–mysql_database=fclusterfs
–volume=74a8f0f627cc0dc6
–audituser='Investigator Name'
/home/user/Desktop/fsmount
```

Multiple file systems can be mounted on the user's host system and multiple SQL servers can provide storage for FClusterfs file-system databases.

Functional overview – dataflow

Having established the component parts of FCluster we can now demonstrate its operation by following data as it is gathered and passed into the system.

The initial imaging process has three deliverables:

1 a SIP containing directory metadata.
2 a collection of SIPs, one each for each file that falls into a 'high value' criteria set by the image acquirer.
3 a conventional 'forensic image', for reference and later extraction of further data.

The selection of files to be packaged as SIPs takes a prioritised triage approach collecting only file types expected to have a higher likelihood of containing evidence depending on the case type.

The first stage of ingestion into FCluster is when the SIP, containing the data defining the file system directory, is

imported into the MySQL database at the heart of FClusterfs. At this stage a directory skeleton will exist but no data is available within FCluster.

The file data, in the form of a number of SIPs, is imported as it becomes available. This starts a process of 'filling out' the evidence file system with data associated with each directory entry. The data is distributed across the Datanodes according to a load balancing algorithm which bases its allocation on benchmarking previously created by running a known set of approved programs against typical data files.

When a SIP arrives on its storage host, it is unpacked and its contents are verified in a number of ways. Only if it is proven to be valid is it then accepted and made available via the distributed file system, FClusterfs. Upon approval at its storage location, a defined list of tasks is invoked and automatic process is conducted, for example generating text indexing or thumb-nailing images.

To provide redundancy and secondary load balancing, a replication agent firstly ensures constant and routine validation of data by applying an SHA 1 checksum to each file; it can then ensure that there are multiple copies of the data, normally three, held on separate hosts within the cluster.

The SIPs at image time will have, most likely, captured only part of the evidence. Subsequently a 'Bag it on demand' system can trigger an on-the-fly acquisition of data that was initially deemed of secondary interest within the image once it has been completed and is available to the cluster. This data is validated and placed in the same assured manner as the rest of the system.

How FCluster is configured as a network system is up to the administrator but it can form a local or wide area network. The prototype successfully uses a VPN to connect the nodes. We've extended it to use nodes on Amazon Web Services. Whenever data is transferred between nodes is it always in an encrypted form and so can be considered safe in a technical sense but this may not be acceptable on principle within a legal environment. The primary objective, and the core of any speed improvement, is that processing takes place locally on the datanode holding the data. In a similar way to the use of SHA1s to identify 'Bad' files, the system can be used without the actual files being accessed. Results are transferred across the network but not normally the data.

FCluster by stages of assurance

FCluster has 4 'zones' of assurance as shown in Fig. 7. We now step through then in more detail.

Acquisition assurance

The first assurance in the system is one of the "Loops of Authority and Acknowledgement" type in which authority is granted to an imaging device to take an image and then FCluster only accepts data that was gathered with that authority Fig. 6.

The FCluster administrator generates an 'Authority to image' in the form of a file which will be issued to a specific device. This file contains a reference number and a randomly generated key which will be used at acquisition time to encrypt the data stored in the SIPs. The reference

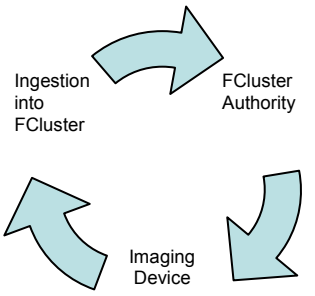

Fig. 6. Acquisition assurance.

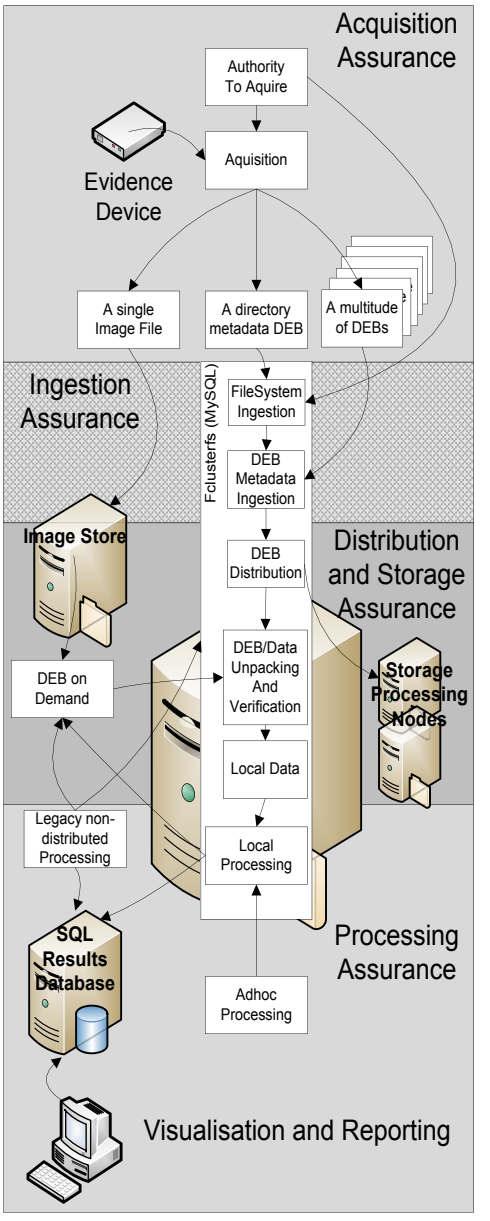

Fig. 7. The 4 zones of assurance.

number and key are recorded in the VolumeInformation table in FClusterfs. Multiple keys can be created and issued to multiple imaging devices to form a 'stock of authorities' to be used over a period of time, the keys have an 'expiry date' associated with them as an added control.

As previously explained, in section 13, the imaging process has three outputs. The SIP containing file-system metadata is the first to be imported. The reference number under which the cryptographic key was recorded is located in VolumeInformation table. If it is present, has not expired or has not been previously fulfilled, the import can proceed. The contents of the SIP is read and the directory meta-data is decrypted and records for each file are created in the inodes, tree and metadata tables. These include fields that describe the full path and filename, file size, MAC dates and times etc. If the key is not present in the VolumeID table, the import cannot proceed.

At the end of this process a complete 'framework' of the directory structure and filenames will have been created in the FClusterfs database. It is actually possible to mount this FClusterfs structure and traverse the directory but as the import of file SIPs that contain file data has not been carried out there is no actual data to analyse in the files.

Ingestion assurance

We now use a series of "checklists" to control the import of the details and contents of the data-file SIPs.

The SIP staging directory, where SIPs are placed ready to be imported, is scanned and any SIPs which form part of a Volume that is expected to be imported are found and the header is read to extract details of the VolumeID, path, filename and size. The inodes table of FClusterfs is searched to see if this SIP is expected, ie there is an entry previously made by a file-system SIP import. At this stage, various fields like the original file's SHA1 and staging directory url should be empty. If there is a record that satisfies these criteria then the fields in the inodes table are populated with the meta-data extracted from the data-SIP. If there is a record in the inodes table and it shows it has already been imported it will not be considered again.

Distribution assurance

This stage has three components. Load Balancing, Moving SIPs to their primary destination and unpacking them.

Load balancing

Having ingested the volume directory metadata the system is now primed to expect the SIPs of data that makeup that file system. The selection of the primary storage of the data is the first task of the ***loadbalancer***. It allocates a storage server to hold the data held within the SIP and records this in the FCluster inodes table. Allocation is based on the available capacity of the host, its processing power and its estimated time to finish its current task list.

The movefile daemon

The movefile daemon also uses "checklist" type assurance by constantly scanning the inodes table of FClusterfs for any SIP that has been allocated a datanode, not been marked as being 'in place' and where the evidence SIP is staged in a local directory. If these conditions are met the SIP is transferred to the storage datanode as allocated by the loadbalancer. If, and only if, the transfer is successful does movedata update the inode table with 'primarystorageinplace' set to true. Movedata is the only mechanism whereby actual data can be moved around the system. It can only operate when all the preconditions from Ingestion Assurance are met. It does not simply scan an evidence folder and move whatever SIPs are present; it moves only expected SIPs, as recorded in the FCluster inodes table, from a folder.

The unpack daemon

Unpacker daemon constantly scans the inodes table to see if there are any SIPs that are on their local server but not unpacked. It takes the entry from the database and looks to see if the files are on its ftp host, as should be the case from the entries in inodes, not the other way round. A file that simply arrives on the server without an entry in inodes would be ignored. When a suitable SIP is identified it is split into header and data sections. The header, containing the metadata is inserted into the 'meta_data' table and the header file erased. The data section is undecoded and the data decrypted with a key stored in the VolumeListing table. This was the key first created and issued by the FCluster and used to encrypt the data in the SIP at acquisition time. If the key does not work, the file cannot be decrypted and so unpacking would fail. Only if the file decrypts and the resulting file has an SHA1 checksum that matches both the name of the file itself and the SHA1 as recorded in the inodes table is the datafile finally accepted.

Processing assurance

The task daemon scans the tasks table to see if any job is required for a file that it holds locally. Because all file access must take place by utilising the enhanced FClusterfs filesystem the file must be the correct file and must have the original content that was collected at imaging time. FClusterfs also gives us fine grained access control to the files within a file system. We could, if we wished, control which users can process specific data with specific programs.

Conclusions

We have demonstrated that by ensuring a rigorous protocol when importing SIPs into a distributed cluster we can provide a level of assurance in data transfer and storage. Additionally, by adopting the same approach as Hadoop we have created a prototype of a middleware specifically designed to address the assurance requirements required in the legal process while providing effective distributed processing. As to whether this does achieve an acceptable level we offer this design for further debate. It should be clear that this design draws upon knowledge from many domains and so there is no single criteria set that can be applied.

Speed concerns

A primary concern with FClusterfs is speed but in practice this has not proven to be a significant problem. Firstly, file

access in existing systems is often across a network connection via SMB and NFS shares. FCluster does this in the same way but using the ftp protocol. These are roughly equal or perhaps slightly slower. Secondly, as we have made clear, FCluster is read-only and so has no record or file locking code. As a result, even when FCluster draws from a remote ftp server data is cached locally in RAM and never needs to refer to the source for updates or changes. Thirdly, the system is designed so that each storage host should process its own local data, so the network issue completely disappears.

All distributed systems suffer from a management overhead. This management issue exists in single host solutions but is exacerbated when management data has to be passed in messages across relatively slow network connections rather than using local memory. This limits scalability but in our initial test we find that the effectiveness of clusters of about 50 hosts on a local Gigabit network does not degrade significantly.

As of Spring 2014, the FCluster prototype is almost complete and we are starting full assessment. We intend this to be available when complete via www.fcluster.org.uk.

Future work

There are many areas in this design that present the opportunity for further research. Our own priorities would include rearranging the database structures to implement the principle of division into subsystems so that the assurance subsystems are reflected in the arrangement of data within the tables. On network security the use of ftp, only used as a protocol for ease when building a prototype, should be replaced with, for example, SSH and use digital certificates as authentication. Issues of Governance and Chain of Custody need to be assessed including comparisons with standards like ISO 27037 and OAIS. The design was always intended to allow existing, legacy, software to run without alteration. We need to consider how we can achieve data abstraction above the middleware.

Acknowledgements

This work is part-funded by the European Social Fund (ESF) through the European Union's Convergence programme administered by the Welsh Government.

References

Abramson D, Foster I, Giddy J, Lewis A, Sosic R, Sutherst R, et al. The Nimrod computational workbench: a case study in desktop metacomputing. Australian Computer Science Communications 1997;19:17–26.

ACPO. ACPO good practice guide for digital evidence http://library.npia.police.uk/docs/acpo/digital-evidence-2012.pdf; 2012.

AFFLIB – the advanced forensic format. http://afflib.sourceforge.net/.

Ayers D. A second generation computer forensic analysis system. Digital Investigation 2009;6:S34–42.

Beebe N. Digital forensic research: the good, the bad and the unaddressed. Advances in Digital Forensics 2009;V:17–36. Springer.

Cloudera. http://cloudera.com/; 2014.

Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. In: OSDI '04: 6th symposium on operating systems design and implementation. USENIX Association; 2004.

Dictionary.com. http://dictionary.reference.com; 2014.

FUSE Filesystem using MySQL as storage.http://mysqlfs.sourceforge.net/; 2013.

Flament R. LoggedFS – filesystem monitoring with Fuse http://loggedfs.sourceforge.net/; 2013.

Garfinkel S. Digital forensics research: the next 10 years. Digital Investigation 2010;7:S64–73.

Globus. https://www.globus.org/; 2014.

Hicks T, Kirkland D, Halcrow M. eCryptfs, a cryptographic stacked filesystem for Linux http://ecryptfs.org/; 2013.

ISO 17025:2005. General requirements for the competence of testing and calibration laboratories. ISO; 2005.

ISO 27001:2013. Information technology – security techniques – information security management systems – requirements. ISO; 2013.

ISO 27037:2012. Information technology — security techniques — guidelines for identification, collection, acquisition, and preservation of digital evidence. ISO; 2012.

Justice FBI, U.D. of. Regional computer forensics laboratory annual report for fiscal year 2012 http://www.rcfl.gov/downloads/documents/RCFL_Nat_Annual12pdf; 2012.

OAIS. http://public.ccsds.org/publications/archive/650x0m2.pdf; 2014.

PCI Security Standards Council. https://www.pcisecuritystandards.org/index.php.

Pringle N, Sutherland I. Is a grid a suitable platform for high performance digital forensics?. In: The 7th European conference on information warfare and security; 2008.

Richard III G, Roussev V. Digital forensics tools: the next generation. Digital Crime and Forensic Science in Cyberspace; 2006:75–90. Idea Group Publishing.

Richard III G, Roussev V, Marziale L. Forensic discovery auditing of digital evidence containers. Digital Investigation 2007;4:88–97.

Robson BA. CurlFtpFS – an FTP filesystem based on Curl and FUSE http://curlftpfs.sourceforge.net/; 2013.

Roussev V, Richard III G. Breaking the performance wall: the case for distributed digital forensics. In: Proceedings of the 2004 digital forensics research workshop; 2004.

Thain D, Tannenbaum T, Livny M. Distributed computing in practice: the condor experience. Concurrency and Computation: Practice and Experience 2005;17:323–56.

Turner P. Selective and intelligent imaging using digital evidence bags. Digital Investigation 2006;3:59–64.

Weka. Waikato environment for knowledge analysis http://www.cs.waikato.ac.nz/ml/index.html; 2014.

kess
Ysgoloriaethau Sgiliau Economi Gwybodaeth
Knowledge Economy Skills Scholarships