Towards the Forensic Investigation on the Hadoop Distributed File System using RAM analysis

Stuart Laing[1], Robert Ludwiniak[1], Brahim El Boudani[2], Christos Chrysoulas[1], George Ubakanma[2], Nikolaos Pitropakis[1]
[1] School of Computing, Engineering and the Build Environment, Edinburgh Napier University, Edinburgh, United Kingdom
{s.laing, r.ludwiniak, c.chrysoulas, n.pitropakis}@napier.ac.uk

[2] School of Engineering, London South Bank University, London, United Kingdom
{elboudab, george.ubakanma}@lsbu.ac.uk

***Abstract*—The usage of cloud systems is at an all-time high, and with more organizations reaching for Big Data the forensic implications must be analyzed. The Hadoop Distributed File System is widely used both as a cloud service and with organizations implementing it themselves. This paper analyzes the forensic viability of a RAM analysis method for Hadoop based investigations and compared it against targeted process data dumping through the Java heap information. The RAM analysis has been done through string searching and the use of the RAM analysis tool Volatility. This work found that RAM analysis can be a valuable tool for discovering artefacts of deleted resources from a Hadoop cluster but was unable to discover further information such as the block to node mapping. The targeted process analysis managed to provide some partial information about deleted resources and also produce important metadata on the current state of the file system.**

Keywords—cloud systems, RAM Analysis, Java Heap Analysis, Hadoop, HDFS, forensic analysis

I. Introduction

Cloud systems are being used more now than ever before, everyday life now involves the use of these systems, social media, government websites, work infrastructure. It's all stored in the cloud. A recent survey from Rackspace Technology has shown that 51% of respondents had their entire infrastructure in the cloud and 41% said they plan to move more of their infrastructure into the cloud. With this increased demand for compute and storage, it is predicted that by 2025 the cloud will store half of all data [1].

In traditional storage systems, the file is saved in its entirety on a disk partition which will be connected to a single host. This is perfectly suited for personal computers and even some professional use cases but cannot scale to the needs of larger organisations. This is where distributed file systems come in. Distributed file systems are installed across hundreds, possibly thousands of individual nodes, these collections of nodes are called clusters and instead of a file being stored on one node the file is instead fragmented and spread across the entire cluster and even duplicated to ensure redundancy in case of failure. This fragmentation also allows read speeds to increase since the file can be read from multiple disks simultaneously which bypasses the typical disk read bottle neck. One of the driving forces behind the popularity of these systems is "Big Data". One of the difficulties in designing systems for Big Data is that the structure of the data itself can vary, these systems need to be able to store databases, videos, spreadsheets, social media posts, and all of these have to be stored in the same cluster.

This paper covers the distributed file system Hadoop. Hadoop was created in 2006 by the Apache software foundation, its name was taken from one of the creator's child's toy elephants which became a symbol for the technology [2, 11]. It was one of the first pieces of software designed for Big Data and is still in use in industry today [3]. The design mantra of Hadoop was to bring processing to data instead of data to processing. It utilises MapReduce, a parallel programming model for writing distributed applications. Hadoop uses the Hadoop Distributed File System (HDFS) and exists as a layer on top of the hosts own file system which means anyone with access to the host operating system can view the file blocks by simply navigating to the correct directory [2].

When it comes to forensics, some of the main benefits of distributed file systems become issues. The first principal of the ACPO Forensic guidelines states that no action should be taken that changes data [4], this is implemented by imaging shutdown (dead) hosts so as to preserve the data, images are then taken to be analysed at a later date. This works fine for a personal computer or phone but when applied to a Hadoop or Ceph cluster quickly becomes impractical. A cluster may contain petabytes of data which cannot be imaged in a reasonable time frame. It is not possible to image dead systems as when a node is taken off the network the system will redistribute the file blocks that it contained. Disabling the entire network would ensure the data changes as little as possible but would incur heavy costs to the organisation.

As the use of these systems increase so do the attacks against them [5], breaches against these systems can happen and forensic investigators need to be able to gather as much information as possible to assess what damage was done and how to prevent attacks in the future. The investigators need to be able to construct a timeline of events, identify what users were involved, and track actions taken. Some attacks may involve deleting files from the network, the investigator may be able to recover these files using forensic techniques.

One of the larger issues with forensic analysis on distributed file systems is acquisition [10]. Since files are fragmented across the cluster acquiring specific files can prove challenging, using the systems features to access the files may change timestamps and access logs which should be avoided so custom tools may be required that can search the meta data of a node to discover the locations of file blocks. There are legal issues that forensics can encounter, these systems can be spread across multiple data centres and possibly countries which means multiple jurisdictions must be navigated. The nodes may hold the personal information of third parties which privacy protections will restrict the viewing of.

The aim of this work is to compare and evaluate two different techniques for Hadoop investigations. The first technique is the analysis of a RAM capture through methods

such as string searching. The second technique is the analysis of heap memory in the Hadoop processes themselves. This is going to be achieved through the design and implementation of a forensic scenario on a virtual environment. The designed scenario will be used to compare the two approaches. Additionally, research will be conducted into the Hadoop forensic artefacts and how they can be recovered, and what other artefacts exist that contain useful forensic evidence. Work will need to be undertaken to discover what effect the configuration of these systems has on the viability of forensic action.

The rest of the paper is structured as follows: Section II provides a detailed literature review. Section III drive us through the used for the experiments methodology, while Section IV in detail describes the experiments and the analysis of the results, and finally Section V concludes our work and provide insight for future work.

II. Literature Review

Leimich et al. [6] lay out a forensic methodology for Hadoop investigations in A RAM Triage Methodology for Hadoop HDFS Forensics which utilises an initial RAM triage before targeted node imaging. The proposed methodology does not follow the traditional forensic stages linearly but instead moves back and forth returning to previous stages when needed. A forensic investigator following this methodology would begin the process without yet knowing the full extent of the investigation, time and resources would need to be estimated to a greater extent than that of a traditional investigation. They found that after deleting a file from the cluster and then taking an image of RAM it can be analyzed to discover which block IDs were removed. Additionally, the paper identified a "magic number" that can be used to identify the blocks in the memory image. The experiment was undertaken on a Hadoop version 1 cluster and the authors admit that there have been some changes between versions 1 and 2, a similar experiment repeated on Hadoop version 3 is likely to see significant differences in results. After the RAM image was taken the authors analyze the dump using basic string searches, no attempts into further memory analysis were attempted which could prove fruitful if only to limit the search space to the relevant processes. This paper is focused on the creation of the forensic methodology and doesn't delve into detail on the analysis, this project hopes to go further in the analysis and extract more information.

Gao and Li [7] put forward a three level mapping for efficient file extraction in their paper A Forensic Method for Efficient File Extraction in HDFS Based on Three-Level Mapping. They state the importance of discovering the mapping between files and nodes which can aid in the extraction of files. An interesting conclusion that the results show is that the larger the files the less blocks are likely to be recovered, this makes sense since larger files have more blocks, and each block only has a limited chance to be recovered. The results of this project will need to be evaluated not only on a per block basis but on a per file basis as well. The experiment that is undertaken in the paper is done with the 3L mapping previously established, the authors write: "Without 3L mapping, it is difficult to overcome the problems caused by the features of cloud and HDFS to implement file extraction." [7]. This is of specific relevance to the work in this project, any analysis of RAM or processes must either mitigate or solve this issue of mapping. Any proposed solution must be evaluated with this in mind.

Sremack [8] wrote all about Hadoop investigations in his book Big Data Forensics - Learning Hadoop Investigations, the book was written about Hadoop version 2 but much of the information is applicable to modern Hadoop. One of the observations Sremack makes is that Big Data forensics is not a replacement for traditional forensics, it exists to augment traditional investigations for the target. One of the conclusions Sremack makes is that metadata in Hadoop can be less valuable than it would be in a more traditional investigation, often metadata is changed or lost when the information passes through Big Data systems and might not be able to be relied upon. While the book is a great resource for technical information about Hadoop and has very detailed information on practical investigations, RAM analysis as a technique is curiously lacking in the work. Very little information is stated about what information is held in RAM and the possibility of an investigation using RAM analysis is not considered [8]. The lack of coverage of RAM analysis as a strategy may point to it not being practical in real investigations. More can be found in [9,10, 11]. Our work hopes to cover this literature gap and show if RAM analysis is a practical solution for investigations.

Summarizing we can argue the following. Hadoop is a commonly used distributed file system. Often it is deployed in the cloud by a cloud service provider as PaaS, forensic investigations on distributed systems can be a challenge which is magnified when they are being hosted by a third party. The traditional forensic strategy cannot be applied to cloud and with more and more services being migrated to the cloud this issue will only increase in scope.

Any solution proposed for cloud forensics need to take into account these challenges and address them. Leimich, et al. [6] showed that RAM forensics can be a possible solution but does come with some drawbacks in the amount and accuracy of its results. Gao and Li state the importance of discovering the mapping between blocks and the DataNodes that store them [7]. The following section discusses the proposed solution to address each of these concerns.

III. Methodology

This section covers the planning and justifications for the implementation. With the knowledge gained from the literature review it was decided that there will be two separate techniques used. The first approach is duplicating the work that was done in A RAM triage methodology for Hadoop HDFS forensics on the most recent version of Hadoop. Alongside this work an additional approach will be done using a targeted Java heap dump of the relevant processes themselves. Both approaches will be done on both the master node and a slave node to compare the two results.

A. Cluster Design

The cluster is designed with the aim of simulating a real Hadoop cluster but on a smaller scale, the smaller scale is necessary to complete this experiment with the limited resources and time imposed upon it. Each host in the network will be using a Linux distribution to allow Hadoop to run. The cluster itself will not be done using physical machines but rather simulated in a virtual environment, each host is a virtual machine inside of VMWare which are networked together.

The cluster will be made up of one master node and three slave nodes. Having three slave nodes allows for a file block to maintain Hadoop's default replication rate of three. The

master node in the cluster will be functioning as both the NameNode and Secondary NameNode since the small size of this cluster means that a Secondary NameNode is not required for stability. It is a possibility that the existence of the Secondary NameNode on the same machine as the NameNode may alter the results of the RAM capture, this is an acceptable risk as literature review shows that the information stored by the Secondary NameNode is identical to the NameNode.

B. Gathering of Test Data & Creating a Forencic Event

To properly simulate the cluster it must be populated with data. A real Hadoop cluster can have petabytes of data but for the purposes of this project only a handful of large files will be uploaded. With Hadoop's block size of 128MB the test files should be larger than this so that they are split into a number of blocks. The raspberry PI operating system images are a good solution here, the files are large enough to have a few blocks under them but not so large that the minimal test cluster will have issues storing and handling them. A number of images will be used and uploaded to the file system inside various directories and sub directories to further simulate a real scenario.

After the data has been uploaded and moved into its final state all the relevant data about the state of the file system will be extracted and saved, this is so that during Section 4 the results of the experiment can be measured against the real data to evaluate accuracy and coverage. This data includes various Hadoop commands that show the structure of the file system with respect to DataNodes. In order to evaluate the proposed forensic processes they must be put to use. In order to facilitate this a forensic event must be executed on the cluster. This forensic event will involve a user account accessing a host on the network and deleting a number of files from HDFS. List of steps which will be undertaken: a) Logs on to a DataNode on the network through SSH, b) Deletes a number of files from the file system with the additional flag of not saving them to trash, and c) Disconnects from the system.

C. Extracting and Analysing the RAM Image and Heap Dump

After the forensic event has been conducted then the RAM will be captured. Two RAM dumps will be taken, one from the master node and one from a slave node. They will be taken using the tool AVML (Acquire Volatile Memory for Linux) developed by Microsoft for capturing the RAM of Linux hosts. Since each host is a VM running in VMWare it would be possible to extract the RAM directly from VMWare, this was decided against to keep the experiment closer to a real scenario. It is vital that the RAM capture occur before the heap dump is taken. The process of taking the heap dump will change the contents of the system memory which should be avoided when possible. After each dump is taken the file will be transferred out of the host.

Each RAM capture will be analysed using the same steps. The capture will first be analysed using Volatility, this will let us extract information about environment variables and process arguments. The second phase of analysis will be a more primitive string search, by searching for strings in the entire RAM dump we can be sure to not only find current data in RAM but any data that has yet to be overwritten. Volatility was chosen for the initial RAM analysis stage. Volatility is a python-based tool for analysing RAM dumps in a variety of formats and operating systems, it allows us to identify the relevant processes in the dumps and extract forensic information. Volatility is less useful when searching for arbitrary data, in this case it makes more sense to search for byte sequences using custom scripts. Volatility version 2 will be used instead of the more up to date version 3 since the support for Linux analysis is still limited in the newer version.

Hadoop's processes are all Java based, Java supports dumping the heap of a process into a standard format that can be parsed. These heap dumps are primarily designed for debugging and finding memory leaks but can be repurposed as data dumps of all classes and instances of a Java process. For dumping the heap of a Java process the command line tool JCMD will be used. The tool allows for commands to be sent to the JVM, one such command is for dumping the heap from a process into a file. JCMD comes standard with installations of Java so it can be assumed to exist on any host that Hadoop is installed on.

Two heap dumps will be taken during the experiment. The first heap dump is on the master node and of the NameNode process, the second dump will be on a slave node and of the DataNode process. As shown in Section 3.A the nodes will have other processes running but these will not be dumped. The Secondary NameNode process will not contain any additional information that the NameNode does not have, and the processes related to MapReduce are not being utilised in the test cluster.

The heap dump file is in a standard format that can be parsed for relevant information. The heap dump contains all data stored in the heap, for the purposes of this work almost all of it is useless. A script will be constructed that reads the relevant data from the dump and outputs it into a standard readable format. The extract script could extract the data directly from the heap dump file but this would require parsing the format of the file itself, Java already has a tool to solve this problem called JHAT. JHAT reads a heap dump and launches a web server that allows a user to navigate the data and find what they need, it supports OQL (Object Query Language) for complex queries. The extract script can instead be run against the web server itself and remove the need for custom parsing.

IV. Design And Impementation

This section covers the entire experiment process from the creation of the cluster to the recording of results. Fig. 1 shows each step that will be taken and the order.

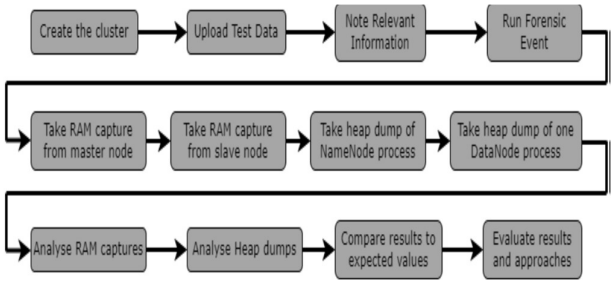

Fig. 1. Flow chart of Experimental Steps

A. Creating the Cluster

As shown in Fig. 2 there are four nodes each running sections of Hadoop. The cluster will be created within VMWare with each node being a virtual machine, each node will then be connected together in a network. Each node is running Ubuntu Server 20.04.3 with 4GB of RAM, 16 processors, and 60GB of disk space. Hadoop communicates

between nodes using SSH, therefore each node needs to have SSH configured to be both client and server. This is also beneficial for ease of use to allow SSHing into the hosts instead of using VMWares interface. Hadoop itself is written and run using Java so each node has Java 1.8.0 342 installed, this installation includes the JCMD tool that is used for the heap dump too. This experiment is conducted on Hadoop version 3.3.1. and then extracted under the directory /opt/hadoop. Each node has the following directory structure showed in the figure below, except that the master-node will not contain the datanode directory and each datanode will not contain the namenode directory.

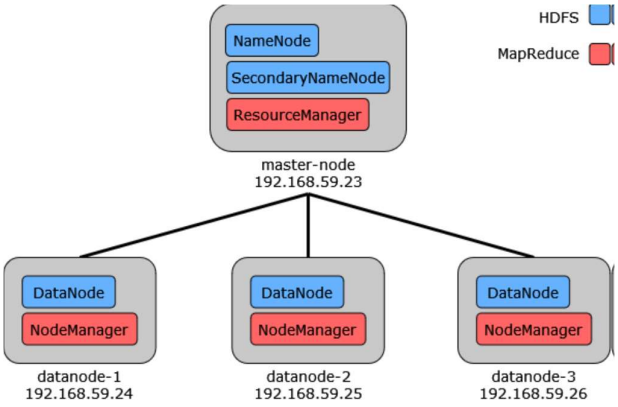

Fig. 2. Topology and Processes of Each Node in the Cluster

After the files have been installed into the correct directories Hadoop needs to be configured. Hadoop has a number of configuration files. Special mention should be made though to three configuration files: a) /opt/hadoop/etc/hadoop/workers, that stores the list of DataNodes and include the three hostnames of the slave nodes and b) /etc/profile and /opt/Hadoop/etc/hadoop/Hadoop-env.sh which configures the environment variables of Hadoop, pointing it to directories and users for the running of Hadoop.

TABLE I. TEST FILES TO BE UPLOADED TO HDFS

File	Size	Obtained From
rockyou.txt	134MB	https://www.kaggle.com/datasets/wjburns/common-password-list-rockyoutxt
PI-OS-64-Bit-Lite.img.xz	289MB	https://downloads.raspberrypi.org/raspios_lite_arm64/images/raspios_lite_arm64-2022-09-26/2022-09-22-raspios-bullseye-arm64-lite.img.xz
PI-OS-Lite.img.xz	338MB	https://downloads.raspberrypi.org/raspios_lite_armhf/images/raspios_lite_armhf-2022-09-26/2022-09-22-raspios-bullseye-armhf-lite.img.xz
PI-OS.img.xz	784MB	https://downloads.raspberrypi.org/raspios_armhf/images/raspios_armhf-2022-09-26/2022-09-22-raspios-bullseye-armhf.img.xz

Once both the NameNode and DataNodes have been configured the cluster can be initialised. Before the cluster can be started each node must be formatted, this is where Hadoop creates files and directories under the ones specified in Figure 9. The DataNodes are formatted when the cluster is started but the NameNode must be formatted separately using the command: hdfs namenode -format. Finally, the cluster can be started. Hadoop includes a script that starts each component individually and provides some useful feedback in case of errors. To check that each node is configured correctly we can check that all the Hadoop processes are running.

B. *Upload the Test Data and Executing the Forensic Event*

Four files were selected to be uploaded to HDFS, the files cover a range of file sizes meaning the number of blocks they are split up into will vary as well. Table I lists each of the test files and the URL that they were obtained from.

The files will be in separate directories within HDFS according to Fig. 3.

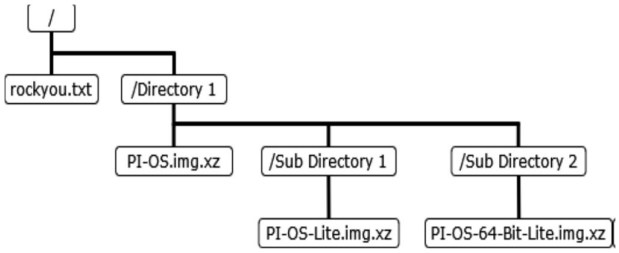

Fig. 3. Example of a figure caption

Hadoop offers a few different methods to upload files to the cluster, after each of the directories are created the files can be uploaded from any of the nodes. Now that the files are uploaded to HDFS the metadata must be extracted to be used in the evaluation. The ***fsck*** command helps here as well, with additional arguments it can show the blocks and their DataNode locations. The full output of the fsck command is all that is needed to evaluate the success of the methods. There are files such as the edits log and fsimage which could be extracted for information but the excess information they provide is of no use to this project and can be ignored. Now that HDFS is in a ready state and the metadata extracted the cluster is ready for the forensic event.

The two files that will be deleted are rockyou.txt and PI-OS-64-Bit-Lite.img.xz. Hadoop has a trash feature for deleted files, in this event the trash will be circumvented using an additional argument in the delete command. The deletion itself will be done from the datanode-3 host using the hadoopuser account. Like how the initial upload was verified using the fsck command the deletion can be verified in the same way. The below listing shows the output after running the fsck command verifying that the files rockyou.txt and PI-OS-64-Bit-Lite.img.xz no longer exist on the file system.

C. *Extracting RAM, Heap Dumps,their Analysis and Discussion*

datanode-1 has been chosen as the DataNode to extract information from along with master-node. Using AVML and JCMD the dumps can be extracted from the hosts for later analysis. This is done immediately following the forensic event in the previous section. Fig. 4 shows the identification of the Hadoop processes along with the creation of RAM and heap dumps for both nodes. After the files were created they were downloaded from the VMs for the analysis in the next.

The Volatility analysis produced minimal results and their interest to an investigator is limited, the second phase of the analysis included string searching through the RAM capture and extracting information. The manual analysis of the captures found promising regex patterns that could be used to find deleted information. The first pattern only recovers information in the NameNodes RAM and the second pattern only finds information in the DataNodes RAM, the patterns do not work across nodes. Each pattern locates the area of memory where the file or block information can be found. In

the NameNodes RAM the file name and path are found in memory after the pattern. When searching through the entire capture multiple entries appear for each file deleted, this is likely due to Java's memory management and can be ignored.

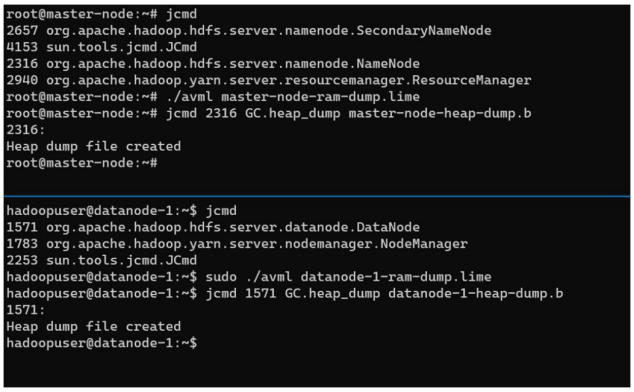

Fig. 4. Extraction of RAM and Heap dumps after the event

The RAM analysis was conducted through string searches and the tool Volatility. The information produced by Volatility was metadata about the cluster itself, the relevant data from both RAM captures is formatted in Table II.

The heap analysis as stated previously was done using a custom script, the output that this script creates is formatted as JSON to allow easy reading and parsing. The DataNode object stores information such as the UUID and hostname. Where multiple objects were recovered such as blocks the information is in a list with each element having further information within. The nature of heap dumps mean that all data held in the process can be extracted, decisions were made about which data was important enough to output into the final JSON file, the output files represent the most forensically relevant information from each of the heap dumps. While the RAM analysis was able to find evidence of the deleted resources the only listed files and blocks in the heap dumps were the non-deleted ones that remained on the system.

TABLE II. INFORMATION GATHERED USING VOLATILITY ON RAM CAPTURES

Information	Value
DataNode User	hadoopuser
NodeManager User	hadoopuser
ResourceManager User	root
SecondaryNameNode User	root
NameNode User	root
Hadoop Conf Dir	/opt/hadoop/etc/hadoop
Master Node IP	192.168.59.23
Data Node 1 IP	192.168.59.24

The fsck command showed that the two files deleted during the experiment were made up of the following blocks (See Fig. 5).

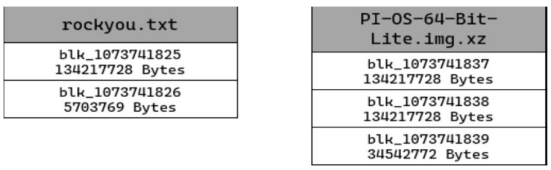

Fig. 5. HDFS Blocks Making up the two Deleted Files

With this information the results can be put into context, since these are the values that are expected to be found in the analysis. In addition to finding the deleted resources the analysis should show metadata about HDFS and the cluster. Using this data, Table III can be constructed to display the results. The Deleted Resource column covers both the files and the underlying blocks. If evidence of the deleted resource was able to be recovered with the technique, then the cell has a yes. Each technique is measured separately, the RAM analysis is split into the results from the NameNode and DataNode, the same is done for the heap analysis.

TABLE III. INFORMATION GATHERED USING VOLATILITY ON RAM CAPTURES

Deleted Resource	NameNode RAM	DataNode RAM	NameNode Heap	DataNode Heap
File: rockyou.txt	Yes	No	No	No
File: PI-OS-64-Bit-Lite.img.xz	Yes	No	No	No
Block: 1073741825	No	Yes	No	No
Block: 1073741826	No	Yes	No	No
Block: 1073741837	No	Yes	No	No
Block: 1073741838	No	Yes	No	No
Block: 1073741839	No	Yes	No	No

Table III demonstrates that the heap analysis was not able to recover any information about the deleted files or blocks. This is consistent with what was learned in the literature review, when files are deleted in HDFS a message is sent to all nodes that store the blocks instructing them to delete their copies, when this happens the objects in Java will be allocated for garbage collection and overwritten. RAM analysis can find the remnants since it takes the entire memory but a heap dump will only extract what Java has not deleted. Additionally as expected the DataNodes have no concept of the HDFS files that the blocks make up and therefore the information was not found using either technique. Interestingly evidence of the block IDs that were deleted was not able to be recovered from the NameNodes RAM capture, despite that this information must be sent to each DataNode.

Alongside the direct information about the deleted resources there is other information such as the file paths or mapping that an investigator would care about. The below table shows the metadata about the resources and cluster and the success of each technique in recovering that information. Table IV is formatted in the same way to the previous table with each technique being split into the node it was done on. Table IV shows that for the metadata both RAM and heap analysis are insufficient to recover the relevant information. Some data such as the filepath for the blocks and files deleted can be recovered. The filepath for the HDFS files is the path inside HDFS while the filepath for the blocks is for the underlying hosts operating system. The DataNode information refers to values such as IP addresses, hostnames, IDs etc. The RAM analysis on the NameNode was unable to recover any DataNode information whereas the heap analysis was able to extract multiple pieces of information about each Node. For the DataNode both the RAM and the heap analysis

were only able to recover the information about the current DataNode itself and not any other node in the cluster. A larger cluster running various jobs may change this due to the larger amount of cross node communication.

TABLE IV. METADATA RELATED TO CLUSTER AND FILE SYSTEM

Metadata	NameNode RAM	DataNode RAM	NameNode Heap	DataNode Heap
File to Block mapping for rockyou.txt	No	No	No	No
File to Block mapping for PI-OS-64-Bit-Lite.img.xz	No	No	No	No
Block to Node mapping for 1073741825	No	No	No	No
Block to Node mapping for 1073741826	No	No	No	No
Block to Node mapping for 1073741837	No	No	No	No
Block to Node mapping for 1073741838	No	No	No	No
Block to Node mapping for 1073741839	No	No	No	No
Filepath for 1073741825	No	Yes	No	Partial
Filepath for 1073741826	No	Yes	No	Partial
Filepath for 1073741837	No	Yes	No	Partial
Filepath for 1073741838	No	Yes	No	Partial
Filepath for 1073741839	No	Yes	No	Partial
Filepath for rockyou.txt	Yes	No	No	No
Filepath for PI-OS-64-Bit-Lite.img.xz	Yes	No	No	No
DataNode Information	No	Partial	Yes	Partial

While the entire filepath for each deleted blocks could be recovered from the DataNodes RAM this information could not be recovered from the heap. However, since the information about non deleted blocks can be recovered it is possible to deduce where in the file system the blocks would have been stored by looking at the locations of the remaining blocks. It is possible that by using the filepaths of the deleted blocks an investigator could recover the block data itself and reconstruct the file, the block IDs are sequential to when they were created which allows the file to be constructed in the correct order by looking at the IDs. Unfortunately, as the number of blocks in a file increases the chance that all blocks will be recovered on a single DataNode drops, meaning that multiple nodes would need to be analysed.

Both results tables do not show the additional metadata that was able to be extracted through the heap dump, while this data is interesting it does not have the same forensic potential as the other values and was chosen to not be included. Four files were selected to be uploaded to HDFS, the files cover a range of file sizes meaning the number of blocks they are split up into will vary as well. Table 1 lists each of the test files and the URL that they were obtained from.

V. CONCLUSIONS

The aim of this work was to compare and evaluate two different techniques for Hadoop investigations. The two techniques that were compared were RAM analysis and Heap analysis. The results from the RAM analysis showed it is possible to recover the file names and paths from the NameNode RAM with the use of a regex pattern. This pattern recovered the file name and path for both files deleted in the forensic event. In the DataNodes RAM with the use of another regex pattern it was possible to find all deleted block IDs and their location on the hosts file system. Both regex patterns were run against the entire RAM capture to retrieve information, with the use of the tool Volatility the specific relevant processes can be analysed to show the environment variables which provide metadata about the cluster. When the Volatility tool was used on the NameNode and DataNode information such as the Hadoop user and configuration directories could be extracted.

The results from the Heap analysis even not the ones expected are still interesting. They showed that even when the heap dump is taken soon after resources have been deleted from the system the relevant instances have already been cleaned by Javas garbage collector. While the instances related to the deleted resources cannot be extracted the remaining files and blocks on the cluster can be extracted, when paired with a historic image of the cluster the missing files can be deduced. While the heap dump was far less effective for recovering the deleted resources other information like the clusters metadata was much greater.

It is clear from the results that while heap analysis is easier and the results more confident it is far less effective in a forensic investigation. With respect to cloud the RAM approach requires less direct interaction with the host than the Heap analysis. The host that is running the NameNode process might not be accessible to the investigator if the system is PaaS which means the CSP will need to be contacted, in this case a RAM image of a host is a relatively simple procedure whereas taking a heap dump requires multiple steps which increases the chance for mistakes if done by a non-forensically trained person.

Further research into this area could be done using a larger cluster, the results seen might not scale when the number of DataNodes exceeds the replication rate of the cluster. Additionally the cluster could be more complex, this could mean more files being stored, MapReduce jobs being run, and various accounts using the cluster concurrent to the experiment taking place. This work was done on a virtual environment simulated using VMWare, this could be improved. The same experiment could instead be ran on a real cloud environment hosted by a CSP. Depending on the kind of cloud environment chosen this may include communication with the CSP itself. One shortcoming of the heap analysis is that it requires commands to be run on the host itself, however the Java processes heap exists in RAM. It should be possible for the entire heap to be extracted not from a running system but a RAM capture.

REFERENCES

[1] Rackspace. “New global rackspace technology survey, in association with google cloud, underscores rapid pace of cloud adoption”. https://www.rackspace.com/newsroom/new-global-rackspace-technology-survey-association-google-cloud-underscores-rapid-pace [Accessed March 2023].

[2] Apache Org. Apache hadoop 3.3.3 documentation. https://hadoop.apache.org/docs/r3.3.3/ [Accessed March 2023].

[3] K.D. Foote. “A brief history of big data”. https://www.dataversity.net/brief-history-big-data/ [Accessed March 2023].

[4] ACPO . “Acpo good practice guide for digital evidence”. https://www.digital-detective.net/digital-forensics-documents/ACPO_Good_Practice_Guide_for_Digital_Evidence_v5.pdf [Accessed March 2023].

[5] B. Thomas. Report shows cyber attacks on cl services have doubled. https://www.bitsight.com/blog/report-shows-cyber-attacks-on-cloud-services-have-doubled [Accessed March 2023].

[6] P. Leimich, J. Harrison, W.J. Buchanan. “A ram triage methodology for hadoop hdfs forensics”. Digital Investigation, Volume 18, 2016, Pages 96-109, ISSN 1742-2876, https://doi.org/10.1016/j.diin.2016.07.003.

[7] Y. Gao, B. Li. “A forensic method for efficient file extraction in hdfs based on three-level mapping”. Wuhan Univ. J. Nat. Sci. 22, 114–126 (2017). https://doi.org/10.1007/s11859-017-1224-7.

[8] J. Sremack. Big data forensics - learning hadoop investigations. Packt Publishing. Retrieved from https://www.packtpub.com/product/big-data-forensics-learning-hadoop-investigations/9781785288104

[9] R. Montasari and R. Hill, "Next-Generation Digital Forensics: Challenges and Future Paradigms," 2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3), London, UK, 2019, pp. 205-212, doi: 10.1109/ICGS3.2019.8688020.

[10] A. Alenezi, H. Atlam, G. Wills. “Experts reviews of a cloud forensic readiness framework for organizations”. Journal of Cloud Computing Advances Systems and Applications, 8 , 14. doi: 10.1186/s13677-019-0133-z

[11] M. Khader, A. Hadi, G. Al-Naymat. “Hdfs file operation fingerprints for forensic investigations”. Digital Investigation, Volume 24, 2018, Pages 50-61, ISSN 1742-2876, https://doi.org/10.1016/j.diin.2017.11.004