Digital Forensics Compute Cluster: A High Speed Distributed Computing Capability for Digital Forensics

Daniel Gonzales, Zev Winkelman, Trung Tran, Ricardo Sanchez, Dulani Woods, John Hollywood

***Abstract*—**We have developed a distributed computing capability, Digital Forensics Compute Cluster (DFORC2) to speed up the ingestion and processing of digital evidence that is resident on computer hard drives. DFORC2 parallelizes evidence ingestion and file processing steps. It can be run on a standalone computer cluster or in the Amazon Web Services (AWS) cloud. When running in a virtualized computing environment, its cluster resources can be dynamically scaled up or down using Kubernetes. DFORC2 is an open source project that uses Autopsy, Apache Spark and Kafka, and other open source software packages. It extends the proven open source digital forensics capabilities of Autopsy to compute clusters and cloud architectures, so digital forensics tasks can be accomplished efficiently by a scalable array of cluster compute nodes. In this paper, we describe DFORC2 and compare it with a standalone version of Autopsy when both are used to process evidence from hard drives of different sizes.

***Keywords*—**Cloud computing, cybersecurity, digital forensics, Kafka, Kubernetes, Spark.

I. Introduction

LAW enforcement agencies (LEAs) and private investigators face an increasing backlog of evidence that must be analyzed to conduct thorough criminal investigations in a wide range of cases. Fig. 1 shows the hard disk drive (HDD) storage capacity of commercially available disk drives has continued grow at an exponential rate from the 1950s to 2016.

Cyber security breach, financial fraud, general criminal, and sexual contraband investigations increasingly require the use of digital forensics. In such investigations it is not uncommon to encounter HDDs with 1 to 2 TB of capacity. Even though HDD capacity growth has slowed from the torrid pace of the early 1990s, HDD capacity is still growing at an exponential rate. By the fourth quarter of 2016 it was possible for consumers to purchase a 10 TB, indicating that LEA investigation backlogs will continue to grow in the future [3]. Furthermore, the situation is likely to become even more challenging than indicated in Fig. 1, as solid state drives (SSDs) can provide even more storage. Seagate claims their Barracuda SSD is the largest consumer storage device available today (in August 2016) and has a capacity of 60 TB [3]. Consequently, LEA and private investigators require faster digital forensics tools. RAND was funded by the National Institute of Justice (NIJ) to develop a distributed computing capability that can accelerate digital forensics analysis. To accomplish this objective RAND developed DFORC2, which is designed to provide LEAs and other private investigators with a cost-effective and efficient digital forensics analysis capability[1]. DFORC2 combines data ingest and file analysis steps so they can be run in parallel instead of serial fashion. We have done this by modifying Autopsy [4], a well-established open source digital forensics tool suite, so it can run on multiple compute nodes in computer cluster. It is designed to run on a compute cluster or in the AWS cloud. DFORC2 on AWS is designed to reduce infrastructure costs, as it will instantiate cloud computing infrastructure only when it is needed, and enables the analyst to tear down the DFORC2 AWS compute cluster when it is no longer needed.

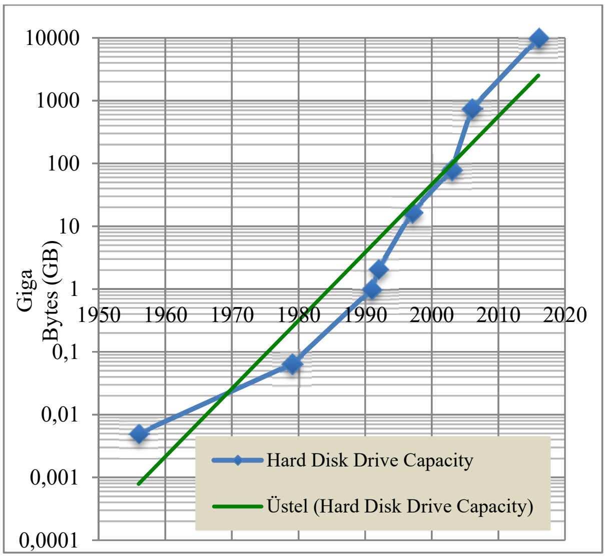


Fig. 1 HDD Storage Capacity Growth [1]-[3]

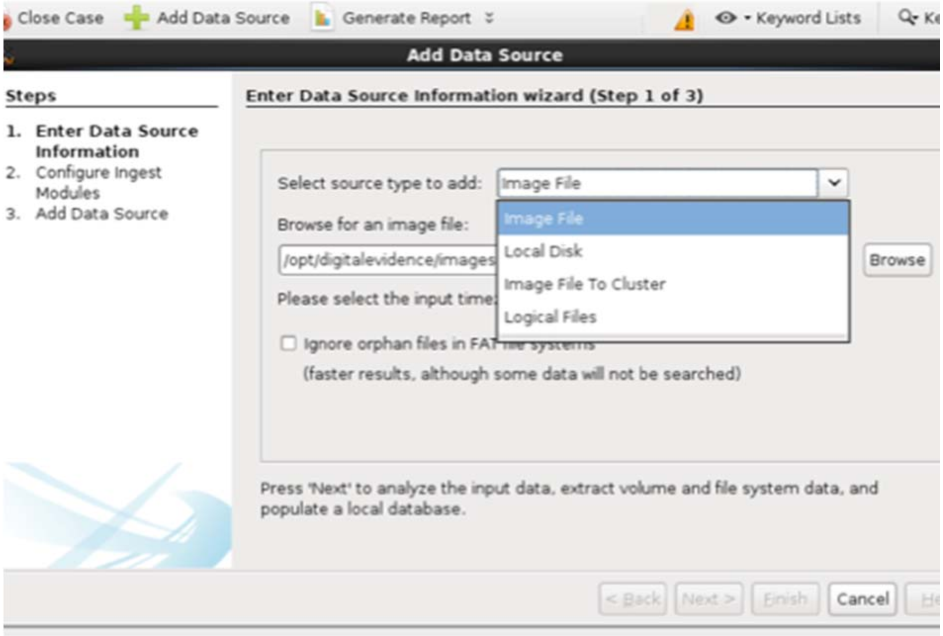
DFORC2 is designed to hide complexity from the user, so it uses the Autopsy User Interface (UI), as shown in Fig. 2.

Users familiar with Autopsy will find the DFORC2 user interface to be very familiar. To start a forensics analysis using

Daniel Gonzales, Zev Winkelman, Trung Tran, Ricardo Sanchez, Dulani Woods and John Hollywood are with the RAND Corporation, 1200 South Hayes St, Arlington, VA 22202 (e-mail: Gonzales@rand.org, zwinkelm@rand.org, ttran@rand.org, rrs@rand.org, johnsh@rand.org, dwoods@rand.org).

[1] This project was supported by NIJ Award No. 2014-IJ-CX-K102, awarded by the National Institute of Justice, Office of Justice Programs, U.S. Department of Justice.

DFORC2, one selects the target where investigation artifacts will be stored. When the user selects “Image file to Cluster,” DFORC2 pipelines are activated and used to ingest and process the subject HDD image.



Add Data Source

Steps

1. Enter Data Source Information
2. Configure Ingest Modules
3. Add Data Source

Enter Data Source Information wizard (Step 1 of 3)

Select source type to add: Image File

- Image File
- Local Disk
- Image File To Cluster
- Logical Files

Browse for an image file:

/opt/digitalevidence/images

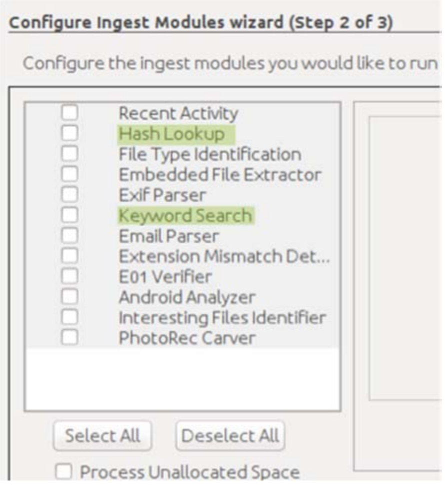
Please select the input time

☐ Ignore orphan files in FAT file systems
(faster results, although some data will not be searched)

Press 'Next' to analyze the input data, extract volume and file system data, and populate a local database.

Fig. 2 DFORC2 Ingest Options

After the image target is selected the user uses the standard Autopsy UI to select which Autopsy forensics modules to run in the forensics processing pipelines, as shown in Fig. 3.



Configure Ingest Modules wizard (Step 2 of 3)

Configure the ingest modules you would like to run

☐ Recent Activity
☐ Hash Lookup
☐ File Type Identification
☐ Embedded File Extractor
☐ Exif Parser
☐ Keyword Search
☐ Email Parser
☐ Extension Mismatch Det...
☐ E01 Verifier
☐ Android Analyzer
☐ Interesting Files Identifier
☐ PhotoRec Carver

☐ Process Unallocated Space

Fig. 3 DFORC2 User Interface

II. DFORC2 System Architecture

The DFORC2 software architecture is shown in Fig. 4. The analysis process starts with ingestion of the subject drive.

A. dc3dd

The disk image is read into the system using the ingest functionality of dc3dd [5]. dc3dd is called through the Autopsy UI as described above. dc3dd also hashes so DFORC2 can confirm each block has not been corrupted during the subsequent processing steps.

B. Kafka Messaging Service

dc3dd sends disk image blocks to the Kafka messaging service [6]. Kafka is a distributed messaging system providing fast, highly scalable and redundant messaging. Kafka is a resilient messaging service designed to support large scale distributed computing systems with a large number of permanent or ad-hoc compute nodes. It is resilient to node failures and supports automatic recovery. By default Kafka employs a reliable messaging protocol where all messages require acknowledgement. Kafka also employs a Publication-subscribe model where producers (in our case dc3dd) send messages (in our case disk blocks) to configurable number of Kafka brokers. Each broker hands a single Kafka partition or message queue, as shown in Fig. 4.

C. Spark

Apache Spark is used to manage the initial stream processing steps involved in image ingest [7]. Each Spark worker node pull disk blocks from a particular Kafka partition. Spark ingest processing nodes are paired with Kafka partitions to maximize performance, as shown in Fig. 4. Data blocks are hashed before and after receipt to ensure integrity. The Spark ingest nodes of DFORC2 identify all "complete" files in each disk block, reconstructs the master logical file system map from these markers, and stores logical file metadata in the Autopsy Postgres database shown in Fig. 4, which is the same database the collaborative configuration of Autopsy uses to hold most investigation case metadata[8].

D. Digital Forensics Worker Nodes

In DFORC2 Spark worker nodes only perform HDD block ingestion and master file system reconstruction tasks. A separate set of digital forensics processing nodes run a specially modified, “headless,” version of Autopsy, so each worker node in this second compute cluster can independently perform digital forensics tasks (e.g., file hashing, digital key word searches, etc.) on logical files found on the HDD, as shown in Fig. 4.

Autopsy uses SOLR, an Apache open source enterprise search platform, to build search string indices [9]. After files are processed by the cluster worker nodes, they are then sent to SOLR, which creates indices to be used in subsequent key word searches.

E. Data Store Options

In contrast to the original standalone version of Autopsy, DFORC2 uses a scalable distributed storage architecture that can be accessed by the Spark and digital forensics processing clusters. When Autopsy runs as a standalone application it stores investigation artifacts in a Postgres database, search string indices are stored in SOLR, and evidence disk blocks and hashes are stored in separate data store on the local file system. In a distributed computing system the local file system should not be used in order to prevent memory conflicts and onerous data synchronization tasks and communications. Therefore, the headless version of Autopsy used in DFORC2 has been modified so it stores disk blocks and hashes in the AWS Elastic File System (EFS) [10] or in AWS Elastic Block Store (EBS) [11]. When DFORC2 is used on a standalone compute cluster it reverts to the local file system used by the server or cluster.

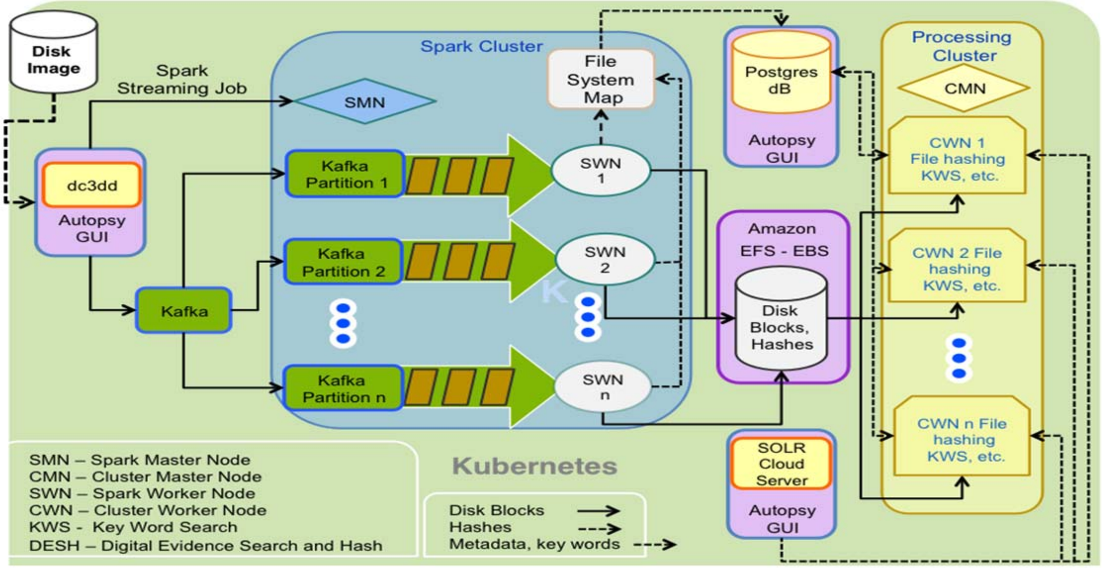


Fig. 4 DFORC2 Software Architecture

Amazon EFS is used because it provides a scalable and cost effective cloud storage solution for digital forensics processing. The size of the EFS volume is scaled up or down automatically by AWS so the user does not have to anticipate how much storage will be required prior to the investigation. In addition, the user does not have reserve a large storage volume that will be costly to reserve and lease. EFS is a cost effective storage solution, except in cases where the number of input/output operations (IOPS) exceeds a certain AWS performance threshold. If the number of IOPS needed exceeds the AWS EFS threshold, EFS IOPS are throttled back to a relatively low rate, which can impact system performance. For these reasons we have experimented with using Amazon EBS instead. EBS has a different pricing structure than EFS and has a higher IOPS performance threshold.

Finally, we note the number of Kafka partitions, Spark nodes and processing cluster nodes are all adjustable by the user. When DFORC2 is run on a stand alone computer cluster, the user will set the number of cluster nodes and Kafka partitions manually. When DFORC2 is run in the cloud these can be set manually, and then the number of digital forensics worker nodes can be adjusted dynamically as described below using the built-in features of Kubernetes.

F. Kubernetes

The underlying foundation of DFORC2 is Kubernetes, which enables cluster size to dynamically scaled up or down [12] in a cloud computing environment. Kubernetes is an open source project that is compatible with several well known cloud computing environments, including AWS. Kubernetes can be used to orchestrate containerized applications and provides auto-scaling so the size of the Spark and processing clusters can be adjusted during run time based on demand. Without Kubernetes the number of Spark and processing cluster worker nodes has to be fixed by the user prior to the start of a run. Without Kubernetes, the digital forensics analysts using DFORC2 would have to estimate the number of Spark and cluster worker nodes needed for a specific size hard disk and for a specific type of investigation. The number of cluster nodes needed could depend on many factors, which may be unknown to the analyst before the investigation. This limitation would likely require the analyst to overprovision the Spark and processing clusters to minimize timely processing of the evidence. Kubernetes solves this problem.

Kubernetes requires applications to be placed in Docker containers. RAND has ported Autopsy and other components of DFORC2 to a standard set of Linux containers. These run as Kubernetes nodes or minions, which can then be linked to a centralized distributed data store, as indicated in Fig. 5.

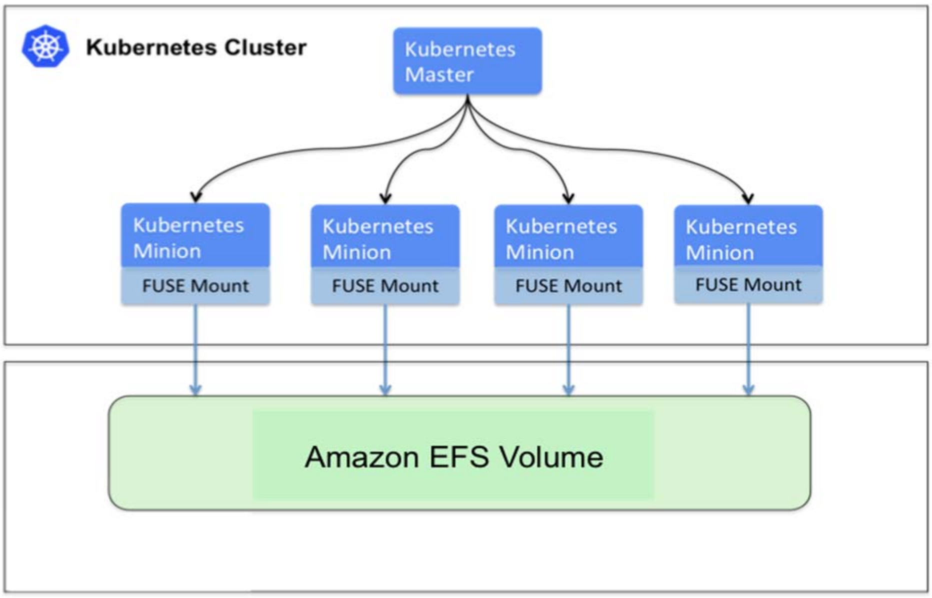


Fig. 5 Kubernetes

III. Standalone Autopsy Performance Tests

First we tested a standalone version of Autopsy using a number of test HDD images of different sizes to determine how fast the stand alone application is when performing standard digital forensics tasks.

A. Test Images

We used the HDD images shown in Table I to evaluate the performance of the stand alone Autopsy application and DFORC2. These test images are widely used in the digital forensics academic community.

TABLE I
Test HDD Images [13], [14]

Image	Size
NPS DOMEX Users, 2009	40 GB
NPS 1weapondeletion, 2011	75 GB
NPS 2weapons, 2011	232 GB
NPS 2 TB, 2011	2 TB

We tested the standard "off the shelf" standalone version of Autopsy when these are run on virtual machines (VMs) of different sizes on AWS. Since the unmodified version of Autopsy is designed to run on Windows, we selected virtual machines that run the Windows Server (2012 release 2) operating system. The test was designed to mirror the image and file processing capabilities of our initial version of DFORC2, so the only Autopsy modules that were enabled were the "Keyword Search" and the "Hash Lookup." Through discussions with the Autopsy developers at Basis Technologies, we learned that these are the most computationally intensive modules to run in Autopsy, and they provide a good basis for a rigorous system performance test. The version of Autopsy we used in testing was the most recent at the time, version 4.1.1. It was configured to run in standalone mode (versus collaborative mode). Standalone Autopsy test results are shown in Fig. 6 for three of the four test HDD images listed in Table I.

Fig. 6 shows the slowest and fastest times Autopsy processed 40, 75, and 232 GB HDD images. Autopsy processing times depend on VM resources. Autopsy was tested on a less capable server with 6.2 ECU and 8 GB RAM (an m4.large AWS instance) and a larger server with 28 ECU and 15 GB RAM (an c3.2xlarge AWS instance) [15].

Stand alone Autopsy test results are not shown in Fig. 6 for the NPS test 2 TB image because it is difficult to compile all of these performance results into a single chart. Therefore, we present stand alone Autopsy test results for the 2 TB test image separately in Table II.

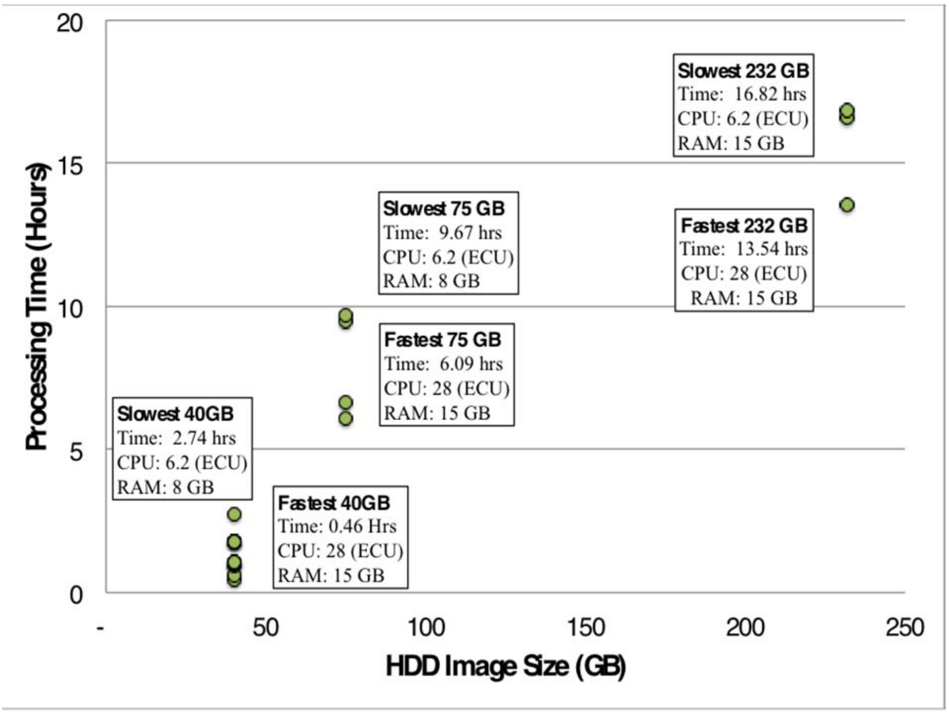


Fig. 6 Stand Alone Autopsy Performance Results

TABLE II
STAND ALONE AUTOPSY TEST RESULTS FOR A 2 TB HDD IMAGE

PROCESSING TIME	AWS VM (ECUs)	Disk IOPS[a]
160 hrs, 37 mins	6.2	300
331 hrs, 39 mins	28	20,000

[a] Input-output operations per second

During these tests we found that local disk performance was a limiting factor on the performance of the stand alone Autopsy application. This limitation became more pronounced for larger HDD images, such as the 2 TB test image. For this test image we found stand alone Autopsy performance could be improved by a factor of two if a high performance disk (a disk capable of 20,000 IOPS) was leased from AWS to support stand alone Autopsy.

The Autopsy test results shown in Fig. 5 do not include the time required to ingest the subject image. Autopsy is not used for image ingestion. Another tool like dc3dd is required to ingest the image from the subject HDD in a forensically sound manner. Total HDD image processing times when using dc3dd and Autopsy together will be the sum of the dc3dd image ingestion and Autopsy file processing times. RAND ingested the test images using dc3dd. dc3dd image ingestion times for 40, 75, and 253 GB size images are shown in Table III.

TABLE III
dc3dd HDD IMAGE INGESTION TIMES

Ingestion Time (minutes)	Image Size
22	40 GB
42	75 GB
142	232 GB

It should also be noted that standalone Autopsy tests were conducted in the AWS cloud, not with a hard disk attached to a server running Autopsy. The test HDD image was resident on the same server used by the standalone Autopsy application. Consequently, any additional communications delays encountered in an actual HDD image ingestion are not replicated in these test results. The same is true of the DFORC2 test results presented in the next section. However, we argue below that such communications time delays are not necessarily the limiting factor in these cases.

IV. DFORC2 PERFORMANCE TESTS

DFORC2 was also tested in the AWS cloud. The DFORC2 test set up was similar to that used for standalone Autopsy, except the test HDD image was placed on a separate server. DFORC2, which was instantiated in a separate computer cluster in the same AWS availability zone, ingested the image from this server.

Fig. 7 presents a Kubernetes dashboard screen shot that shows the HDD image block ingestion traffic into the EFS volume on the DFORC2 EFS server. This traffic is shown by the lighter line in the top pane of the figure. Shown in darker line in the top pane is communication traffic to the EFS volume from the processing cluster worker nodes that perform file processing simultaneously while image ingestion is ongoing. Processing cluster worker node traffic is usually much lower but can spike up occasionally. In the case of the 232 GB image disk block hashing continues after image ingestion and digital forensics file processing is completed (the results shown in Fig. 7 are for the 232 GB image case).

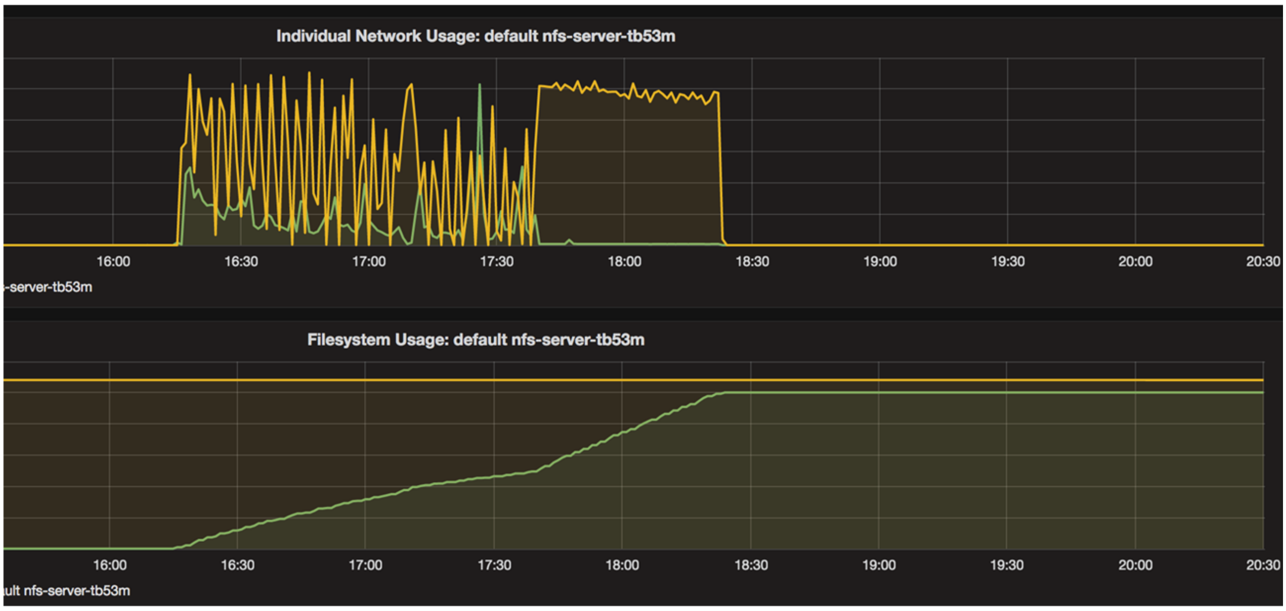


Fig. 7 Kubernetes EBS Volume Node Dashboard

The darker line in bottom pane of Fig. 7 shows how quickly the EFS volume is filled by the Spark streaming process. The image ingestion process is finished in a little over 2 hours for the 232 GB image. Image ingestion in this case is slower than if a dedicated server performed the ingestion process using dc3dd (from Table II we note that dc3dd completed ingestion in 2 hours and 22 minutes).

Five Spark nodes and five Kafka partitions were used in the DFORC2 tests reported here for HDD images that were 232 GB or smaller. For the 2 TB image, ten Spark nodes and Kafka partitions were used. From experimentation with compute clusters of different sizes we have found that DFORC2 image ingestion times can be reduced for large disks by increasing the number of Spark nodes and Kafka partitions.

DFORC2 performance results are shown in Table III. As noted above DFORC2 was configured with 5 or 10 Spark nodes and the same number of Kafka partitions. The number of worker nodes was varied between 10 and 66 nodes. 66 processing nodes was the maximum available in the RAND AWS enclave for this testing when 5 spark nodes were used. This was a limit on AWS enclave permissions, and is not a limitation of DFORC2. All nodes or minions in the Kubernetes cluster used in these tests were standard AWS M4.large instances.

TABLE IV
DFORC2 Test Results

Ingestion & Processing Time (Hours:Min:Sec)	Processing Cluster Nodes	Spark Nodes/Kafka Partitions	HDD Image Size (GB)
0:22:00	20	5/5	40
0:33:00	20	5/5	75
4:56:00	66	5/5	232
22:30:05	35	10/10	2048

For the 40 GB image case, DFORC2 file processing in a little over 16 minutes, while hashing of the ingested files by dc3dd takes longer – or 22 minutes total. With 10 process cluster worker nodes DFORC2 completes processing a 75 GB image in about 28 minutes, while again dc3dd block hashing (which is a single server process) took longer – or 33 mins. In contrast, dc3dd and stand alone Autopsy requires almost 7 hours to complete ingestion and processing of a 75 GB HDD image.

Table IV shows that it took DFORC2 about 4 hours and 56 minutes to process a 232 GB image if it had access to 66 worker nodes, although it took only 2 hours and 56 minutes to complete digital forensic file processing. Again, hashing of disk image blocks by dc3dd took significantly longer.

Fig. 8 compares the performance of DFORC2 to that of dc3dd and stand alone Autopsy. HDD disk image size is shown in the horizontal axis and time in hours is shown in the vertical axis. The results shown include image ingestion and file processing steps for both systems. As mentioned above DFORC2 performs image ingestion simultaneously with file processing. By combining these steps the overall investigation process can be sped up considerably, as shown in Fig. 8. DFORC2 completes image ingestion and file processing tasks much faster than an Autopsy based stand alone server system, even when it is equipped with a high performance disk. DFORC2 cuts the time for processing a 232 DG image by a factor of three.

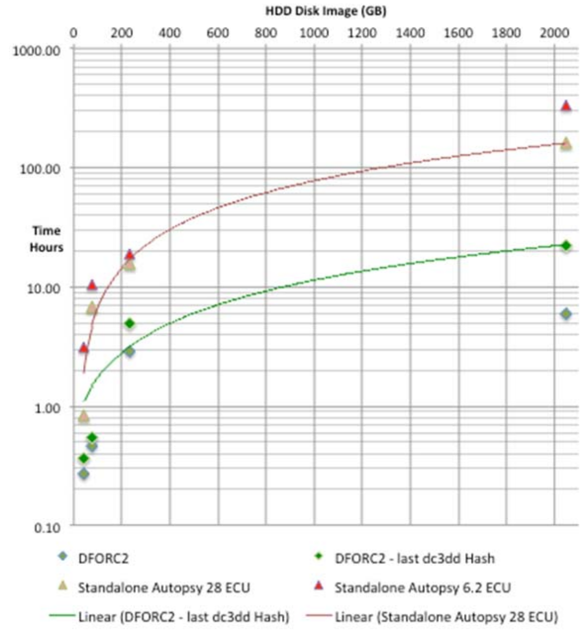


Fig. 8 DFORC2 & dc3dd/Autopsy Performance Comparison

The results shown include image ingestion and file processing steps for both systems. As mentioned above DFORC2 performs image ingestion simultaneously with file processing. By combining these steps the overall investigation process can be sped up considerably, as shown in the figure. DFORC2 completes image ingestion and file processing tasks much faster than the Autopsy based stand alone server system. As mentioned above DFORC2 performance for processing large HDD images is limited by the speed at which dc3dd can hash image blocks. For a 2 TB image this is the last remaining bottleneck in DFORC2 that limits performance, for which it takes a total of 22 hours and 33 minutes to complete both image ingestion and digital forensics file processing. In contrast, with a normal server and local disk, stand alone Autopsy takes about 331 hours to perform the same forensics tasks on the same HDD image. In this case, DFORC2 cuts total processing time by a factor of 14, which is over an order of magnitude improvement in performance, as indicated by the two linear trend lines shown in Fig. 8.

V. Using DFORC2 in Computing Clouds

To simplify the test environment and manage costs we tested both Autopsy and DFORC2 in the AWS cloud. Many LEAs do not use cloud computing services to support digital forensics investigations because they lack tools that work effectively in the cloud, but also because of legal concerns as to whether the chain of custody for digital evidence can be demonstrated and maintained effectively in computing clouds. While this subject is beyond the scope of this paper, it is a subject that RAND is examining for DFORC2 (the results of this investigation will be reported in a separate paper).

DFORC2 has been designed so it can be run on a stand alone computer cluster, although in this deployment case, the number of Spark and processing cluster worker nodes will be limited by the size of the on-premise cluster.

DFORC2 offers the most significant performance improvement for digital forensics when it is used as a cloud based application. There are four ways DFORC2 can be used in the cloud. These along with the traditional stand alone server deployment approach are shown in Table IV.

TABLE V
DFORC2 Implementation Options

Option	Streaming	Distributed	Cloud
DFORC2 on premise single machine	Yes	No	No
DFORC2 on premise data center	Yes	Yes	No
DFORC2 on premise – remote data center	No	No	Yes
Ship HDDs to AWS DFORC2 processing	Yes	No	Yes

The traditional approach for using DFORC2 is shown in the second of Table V. In this case the HDD image would be ingested locally on a single machine or server. In the traditional approach, DFORC2 cannot be run as a distributed application, but with a multi-core server it would be run with its Spark streaming ingest capability. So, in this case, it is possible that DFORC2 could provide some performance improvement over using the combination of dc3dd and Autopsy for the same investigation. DFORC2 would also provide the advantage that data acquisition and analysis can be done at the same time, as the analyst would not have to wait several hours or days for the image ingestion job to complete.

The second option shown in the table is to ingest the HDD image locally on premise at a private datacenter that offered multiple servers. In this case DFORC2 could be loaded on multiple servers and multiple Spark and processing cluster nodes could be instantiated, but the full functionality of Kubernetes (e.g., dynamic scaling) may not be available.

The third option shown in the table is to acquire the HDD image on premise and to stream the drive disk blocks into the Spark cluster over a high speed communications link to a remote data center or cloud where DFORC2 would run. In this case the HDD image would be acquired locally, but ingested remotely (e.g., in the cloud). Spark is designed to support this approach, but it should be noted that good communications link with a bandwidth at least as high as dc3dd ingest speed is needed to make this approach practical. RAND has measured dc3dd ingest speed over USB 3.0 and determined it to be approximately 15 Mbps, which is therefore maximum ingestion speed for HDD images using common computing hardware.

The fourth option is to ship HDD evidence to a cloud service provider for remote acquisition, ingestion, and file processing. This option is only desirable if a high speed communications link to the remote cloud data center is not available. We plan to investigate feasibility of employing this option with AWS.

VI. Conclusions

In this paper we have described the design and capabilities of a new open source digital forensics tool, DFORC2, which leverages state of the art distributed computing capabilities being used in a variety of industries. DFORC2 performs image ingestion and logical file processing tasks simultaneously to speed up the analysis of digital evidence.

We evaluated the performance of DFORC2 and compared it to a standalone version of Autopsy. The results demonstrate DFORC2 is substantially faster than standalone Autopsy, and that this performance advantage increases as the size of the subject HDD image increases. While the chain of custody for such a system remains to be demonstrated, its appeal is clear.

In future work RAND plans to demonstrate that a high integrity chain of custody can be established for DFORC2, using the features of its components and the high integrity computing and storage features available from leading cloud computing service providers, like AWS.

Acknowledgment

The authors wish to help Adrian Salas and Cord Thomas, of RAND, for their invaluable assistance in reconfiguring RAND firewalls and RAND AWS enclave authentication and access control settings to enable this research.

References

[1] S. J. Vaughan-Nichols, “Hard drive technology reaches a turning point,” *Computer*, vol. 36, no. 12, pp. 21–23, 2003.

[2] “Timeline: 50 Years of Hard Drives,” *PCWorld*, 13-Sep-2006. (Online). Available: http://www.pcworld.com/article/127105/article.html. (Accessed: 04-Apr-2017).

[3] “Seagate’s 10TB Barracuda Pro is the world’s largest consumer hard drive,” *PCWorld*, 19-Jul-2016. (Online). Available: http://www.pcworld.com/article/3096292/storage/seagates-10tb-barracuda-pro-is-the-worlds-largest-consumer-hard-drive.html. (Accessed: 04-Apr-2017).

[4] “Autopsy.” (Online). Available: http://www.sleuthkit.org/autopsy/. (Accessed: 28-Jan-2016).

[5] “dc3dd download | SourceForge.net.” (Online). Available: http://sourceforge.net/projects/dc3dd/. (Accessed: 27-Jan-2016).

[6] “Apache Kafka.” (Online). Available: http://kafka.apache.org/index.html. (Accessed: 09-Jun-2015).

[7] “Apache Spark^{TM} - Lightning-Fast Cluster Computing.” (Online). Available: https://spark.apache.org/. (Accessed: 09-Jun-2015).

[8] “PostgreSQL: The world’s most advanced open source database.” (Online). Available: http://www.postgresql.org/. (Accessed: 28-Jan-2016).

[9] “Apache Solr -.” (Online). Available: http://lucene.apache.org/solr/. (Accessed: 05-Apr-2017).

[10] “Amazon EFS Performance - Amazon Elastic File System.” (Online). Available: http://docs.aws.amazon.com/efs/latest/ug/performance.html. (Accessed: 30-Jan-2017).

[11] “Amazon Elastic Block Store (EBS) – Block Storage for EC2,” *Amazon Web Services, Inc.* (Online). Available: //aws.amazon.com/ebs/. (Accessed: 30-Jan-2017).

[12] “Kubernetes,” *Kubernetes*. (Online). Available: http://kubernetes.io/. (Accessed: 30-Jan-2017).

[13] “Digital Corpora.”.

[14] “The CFReDS Project.” (Online). Available: https://www.cfreds.nist.gov/. (Accessed: 05-Apr-2017).

[15] “Amazon EC2 FAQs - Amazon Web Services,” *Amazon Web Services, Inc.* (Online). Available: //aws.amazon.com/ec2/faqs/. (Accessed: 05-Apr-2017).