SCARF

CONTAINER-BASED APPROACH TO CLOUD-SCALE DIGITAL FORENSICS
ABOUT US

- University of New Orleans
- GNOCIA
  - Greater New Orleans Center for Information Assurance
- Chris Stelly
  - cd@stelly.org  @cdstelly
- Vassil Roussev
  - vassil@roussev.net
OVERVIEW

- Motivation for building SCARF
- Previous work
- SCARF goals
- SCARF design
- SCARF architecture and walkthrough
- Results of testing
MOTIVATION

- Massive amounts of data to investigate, only getting worse
- As SSD, NVMe, and other speedy drives become popular, current generation of tools won't perform at speed
  - Spinning disks have masked a processing bottleneck
- Current forensic tools aren't built for scale
### VARIOUS DISKS FOR SALE (AUG 2017)

<table>
<thead>
<tr>
<th>Spinning Disks</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Size (TB)</td>
<td>$/TB</td>
</tr>
<tr>
<td>199.99</td>
<td>8</td>
<td>24.99</td>
</tr>
<tr>
<td>109.99</td>
<td>4</td>
<td>27.50</td>
</tr>
<tr>
<td>89.99</td>
<td>3</td>
<td>30.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solid State</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Size (GB)</td>
<td>$/TB</td>
</tr>
<tr>
<td>274.99</td>
<td>1024</td>
<td>274.99</td>
</tr>
<tr>
<td>179.99</td>
<td>512</td>
<td>359.98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NVMe</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Size (GB)</td>
<td>$/TB</td>
</tr>
<tr>
<td>127.99</td>
<td>250</td>
<td>512.96</td>
</tr>
<tr>
<td>199.99</td>
<td>512</td>
<td>399.98</td>
</tr>
<tr>
<td>447.88</td>
<td>1024</td>
<td>447.88</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>USB Drives</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Size (GB)</td>
<td>$/TB</td>
</tr>
<tr>
<td>6.99</td>
<td>32</td>
<td>223.68</td>
</tr>
<tr>
<td>40.99</td>
<td>128</td>
<td>327.92</td>
</tr>
<tr>
<td>61.17</td>
<td>256</td>
<td>244.68</td>
</tr>
</tbody>
</table>
SCARF: SCALABLE REALTIME FORENSICS

OBVIOUS PROBLEM, SO WHAT HAVE OTHERS DONE

- Sleuthkit Hadoop abandoned, but map/reduce isn't a good fit anyway - we want to process streaming data
- Hansken project at Netherlands Forensic Institute not open source
CONTAINERS

- Core idea is not new - *jail*

- We can think of as easy-to-implement, low-cost modules for specific tasks
  - Task examples - hashing, ExifTools, BulkExtractor

- Most importantly: **containers can be SCALED**

- Docker is the choice here - others can be used (rkt, LXD, Kubernetes)
  - Kubernetes was initially used
SCARF: SCALABLE REALTIME FORENSICS

SCARF: GOALS

- **SCA**lable **R**ealtime **F**orensics
- **Top priority** is to *effectively* throw more hardware at any dataset
- Ability to incorporate existing forensic tools
- Stream processing - start get answers before a full target read
- Ultimately, we want to process at IO speed
  - If we can read from SSD at 500MB/s, want to analyze at 500MB/s
SCARF: DESIGN

- SCARF is a *framework* built to support digital forensics
  - Distributed - can throw more hardware at the problem
  - Extensible - can use existing tools (BulkExtractor, Tika)
  - Accountable - keeps track of executed and failed ‘tasks’
  - Retrievable - keeps results in scalable database for real-time queries
- Open Source
Utilize Docker for individual containers

Docker Swarm for scaling

How is distribution achieved?

- As data is streamed in from forensic target (ex. HDD), it will be evenly distributed to Cluster Nodes
- When Cluster Nodes execute tasks, they distribute them to 'Worker' containers
- DNS round-robin for task distribution
- Different workers for different tasks
- Scale up containers based on priority, easily!

```bash
docker service scale exiftools=96
```
DOCKERFILE SAMPLE

1. FROM anapsix/alpine-java
2. MAINTAINER Wurstmeister
3. RUN apk add --update unzip wget curl docker jq coreutils

5. ENV KAFKA_VERSION="0.10.1.0" SCALA_VERSION="2.11"
6. tar xzf /tmp/kafka_${SCALA_VERSION}-${KAFKA_VERSION}.tgz -C /opt
   VOLUME ["/kafka"]
7. ENV KAFKA_HOME /opt/kafka_${SCALA_VERSION}-${KAFKA_VERSION}

9. ADD start-kafka.sh /usr/bin/start-kafka.sh
10. ADD broker-list.sh /usr/bin/broker-list.sh
11. ADD create-topics.sh /usr/bin/create-topics.sh

13. # The scripts need to have executable permission
14. RUN chmod a+x /usr/bin/start-kafka.sh && \
15.    chmod a+x /usr/bin/broker-list.sh && \
16.    chmod a+x /usr/bin/create-topics.sh

18. CMD ["start-kafka.sh"]
SCARF: SCALABLE REALTIME FORENSICS

SCARF: DISTRIBUTED EXTENSIBLE ACCOUNTABLE RETRIEVABLE

- Easily design new containers for forensic tools
- We call it 'dockerizing'
- Idea is to wrap forensic tools with a container, and expose an RPC method
  - RPC method takes raw data as a parameter, yields result as JSON
- Implemented ‘dockerization’ of Yahoo's OpenNSFW in a matter of minutes
Actual ExifTool invocation:

```
func (t *RPC) Execute(args *Args, reply *string) {
  toolPath := "/usr/bin/exiftool"
  // Setup the shell command to launch ExifTool
  opts := []string{"-"}
  cmd := exec.Command(toolPath, opts)
  cmd.Stdin = bytes.NewReader(args.Data)
  var out bytes.Buffer
  cmd.Stdout = &out
  err := cmd.Run()
  // - debug output
  fmt.Println(out.String())
  *reply = out.String()
}
```
At large scale we can expect to encounter errors frequently. So, we need to track each task to ensure completion.

- We utilize **Apache Kafka** to track issued and completed tasks.
  - Ex: {taskID: 5, taskName: MD5, fileID: 89500}

- Kafka itself is distributed, avoiding a single point of failure.
SCARF: SCALABLE REALTIME FORENSICS

KAFKA AND MESSAGE LOGS

Anatomy of a Topic

Partition 0
0 1 2 3 4 5 6 7 8 9 0 1 1 2

Partition 1
0 1 2 3 4 5 6 7 8 9

Partition 2
0 1 2 3 4 5 6 7 8 9 0 1 1 2

Old → New

Producers

writes

0 1 2 3 4 5 6 7 8 9 1 1 1 2

read

Consumer A (offset=9)

Consumer B (offset=11)
At scale, we can expect to have a large results dataset

ElasticSearch provides a scalable, distributed database

- Again, allows us to throw more hardware at the problem while being resilient to failure

Importantly, ElasticSearch distribution allows us to query indexed data with *minimal response time*

- Schema-less allows flexibility
ELASTICSEARCH - INSERT TO DATABASE

```json
PUT /scarf/file/85000
{
  "id": 85000,
  "filenames": ["Lord-of-the-Flies.pdf"],
  "Createtime": 1501875401,
  "Modifytime": 1501875401,
  "Accessstime": 1501875401,
  "Emodifytime": 1501875401,
  "Fflags": "",
  "Flags": "",
  "Filesize": 285166,
  "Dataruns": []
}
```
SCARF: SCALABLE REALTIME FORENSICS

SCARF: DESIGN

- SCARF is a *framework* built to support digital forensics
  - **Distributed** - Docker Swarm
  - **Extensible** - Docker Container + RPC method
  - **Accountable** - Apache Kafka
  - **Retrievable** - ElasticSearch
- Open Source
ARCHITECTURE - HOW DO THE PIECES FIT TOGETHER?

- Task - operation on a specified file, e.g. SHA1 on 'flower.jpg'
- Broker - handles reading of forensic data
- Worker - container for a task, e.g. BulkExtractor container
- Task Manager - coordinates task logs
  - Logs - used to track individual 'tasks' sent to workers
- ElasticSearch - stores metadata of forensic target and any results from workers
SCARF: SCALABLE REALTIME FORENSICS
SCARF: SCALABLE REALTIME FORENSICS

DATA BROKER

forensic target

metadata parser

bulk streamer

file streamer

MD5: "flower.jpg"

Cluster: 1, FileID: 85000, Task: MD5

TASK MANAGER

FileID: 85000, MD5: 50a..20

WORKER POOL

ELASTICSEARCH SERVICE

Cluster Node

Cluster Node

Cluster Node
SCARF: SCALABALE REALTIME FORENSICS

STATUS

- Supports NTFS images
- Broker: Raw, File-based
- Workers:
  - MD5
  - SHA1
  - Apache Tika
  - Yahoo OpenNSFW
  - ExifTools
  - BulkExtractor
Testing It Out

- **Dataset:**
  200GB from govdocs corpus

- **Older Server:**
  4 nodes
  24 cores each
  256GB RAM each

- **96 total CPUs**
### NUMBERS

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>8</th>
<th>32</th>
<th>48</th>
<th>96</th>
<th>192</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SHA1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># containers</td>
<td>4</td>
<td>8</td>
<td>32</td>
<td>48</td>
<td>96</td>
<td>192</td>
</tr>
<tr>
<td>MB/s</td>
<td>345</td>
<td>857</td>
<td>924</td>
<td>985</td>
<td>948</td>
<td>992</td>
</tr>
<tr>
<td><strong>OpenNSFW</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># containers</td>
<td>4</td>
<td>8</td>
<td>32</td>
<td>64</td>
<td>96</td>
<td>192</td>
</tr>
<tr>
<td>MB/s</td>
<td>0.4</td>
<td>1.4</td>
<td>3.8</td>
<td>7.2</td>
<td>10.9</td>
<td>21.3</td>
</tr>
<tr>
<td><strong>ExifTools</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># containers</td>
<td>4</td>
<td>8</td>
<td>32</td>
<td>64</td>
<td>96</td>
<td>192</td>
</tr>
<tr>
<td>MB/s</td>
<td>5.2</td>
<td>17</td>
<td>99</td>
<td>151</td>
<td>170</td>
<td>192</td>
</tr>
<tr>
<td>Per container</td>
<td>1.3</td>
<td>2.1</td>
<td>3.1</td>
<td>2.4</td>
<td>1.8</td>
<td>1</td>
</tr>
</tbody>
</table>
## NUMBERS

<table>
<thead>
<tr>
<th></th>
<th>Tika</th>
<th>BulkExtractor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># containers</td>
<td>4</td>
<td>12</td>
<td>24</td>
</tr>
<tr>
<td>MB/s</td>
<td>0.5</td>
<td>1.1</td>
<td>2.4</td>
</tr>
<tr>
<td>Per container</td>
<td>0.13</td>
<td>0.09</td>
<td>0.10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>12</th>
<th>24</th>
<th>48</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB/s</td>
<td>3.5</td>
<td>17.9</td>
<td>22.7</td>
<td>54.8</td>
<td>59.4</td>
<td>151.5</td>
</tr>
<tr>
<td>Per container</td>
<td>0.9</td>
<td>1.5</td>
<td>0.9</td>
<td>1.1</td>
<td>0.9</td>
<td>1.2</td>
</tr>
</tbody>
</table>
SCARF: SCALABLE REALTIME FORENSICS

SCREENSHOTS

```json
PUT /scarf/file/85000
{
  "id": 85000,
  "filenames": ["Lord-of-the-Flies.pdf"],
  "Createtime": 1501875401,
  "Modiftytime": 1501875401,
  "Accessstime": 1501875401,
  "Emodifytime": 1501875401,
  "Fflags": "",
  "Flags": "",
  "Filesize": 285166,
  "Dataruns": []
}
```
Searching for all PDFs
SCARF: SCALABLE REALTIME FORENSICS

SCREENSHOTS

EXIFTool Results

OpenNSFW
NEXT STEPS

▸ Improve operator ease-of-use (with GUI)

▸ Increase size of forensic targets

▸ Investigate Tika performance bottleneck

▸ Deploy to AWS/Azure

▸ Investigate security and throughput implications

▸ Smarter distribution among workers (currently round robin)
SCARF: SCALABLE REALTIME FORENSICS

SUMMARY

- SCARF is a framework designed to scale to demands of digital forensic investigations
- Incorporate existing tools!!
- On an older, 4-node cluster, tests show increased overall throughput with an increased number of containers
  - BulkExtractor up to 150MB/s
- Newer 4-node cluster shows significant increase in throughput
  - 180% increase for ExifTools throughput, 80% for Tika, and 64% for OpenNSFW
QUESTIONS?