

CHRISTOPHER STELLY

VASSIL ROUSSEV, PHD

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)

Spinning Disks

\$	Size (TB)	\$/TB
199.99	8	24.99
109.99	4	27.50
89.99	3	30.00

Solid State

\$	Size (GB)	\$/TB
274.99	1024	274.99
179.99	512	359.98

NVMe

\$	Size (GB)	\$/TB
127.99	250	512.96
199.99	512	399.98
447.88	1024	447.88

USB Drives

\$	Size (GB)	\$/TB
6.99	32	223.68
40.99	128	327.92
61.17	256	244.68

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

The screenshot shows the GitHub repository page for 'sleuthkit/hadoop_framework'. The repository is described as 'This is a prototype system that uses Hadoop to process hard drive images.' It has 404 commits, 2 branches, and 0 releases. The page features a navigation bar with 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', and 'Insights'. A yellow progress bar is visible. Below the navigation bar, there are buttons for 'New file', 'Upload files', 'Find file', and 'Clone or download'. A commit history table is shown, with the latest commit highlighted in blue. The commit history table has the following data:

Commit Message	Time Ago
Latest commit a855c40 on Sep 30, 2012	
bin and comments.	6 years ago
Removed unnecessary JobData class. Added documentat	6 years ago
Added copyright/license block	6 years ago
Added copyright/license block	6 years ago
Comments and documentation in crossdrive-scoring cod	6 years ago
added deploy dir and scripts	6 years ago
Lightbox commit	6 years ago
Added JAR containing native libs for unixsocket. Added	6 years ago
Comments and documentation in match code.	6 years ago
Added copyright/license block	6 years ago
Added copyright/license block	6 years ago

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: 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

SCARF: DISTRIBUTED EXTENSIBLE ACCOUNTABLE RETRIEVABLE

- ▶ 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!
 - ▶ `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: 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

SCARF: DISTRIBUTED **EXTENSIBLE** ACCOUNTABLE RETRIEVABLE

▶ Actual ExifTool invocation:

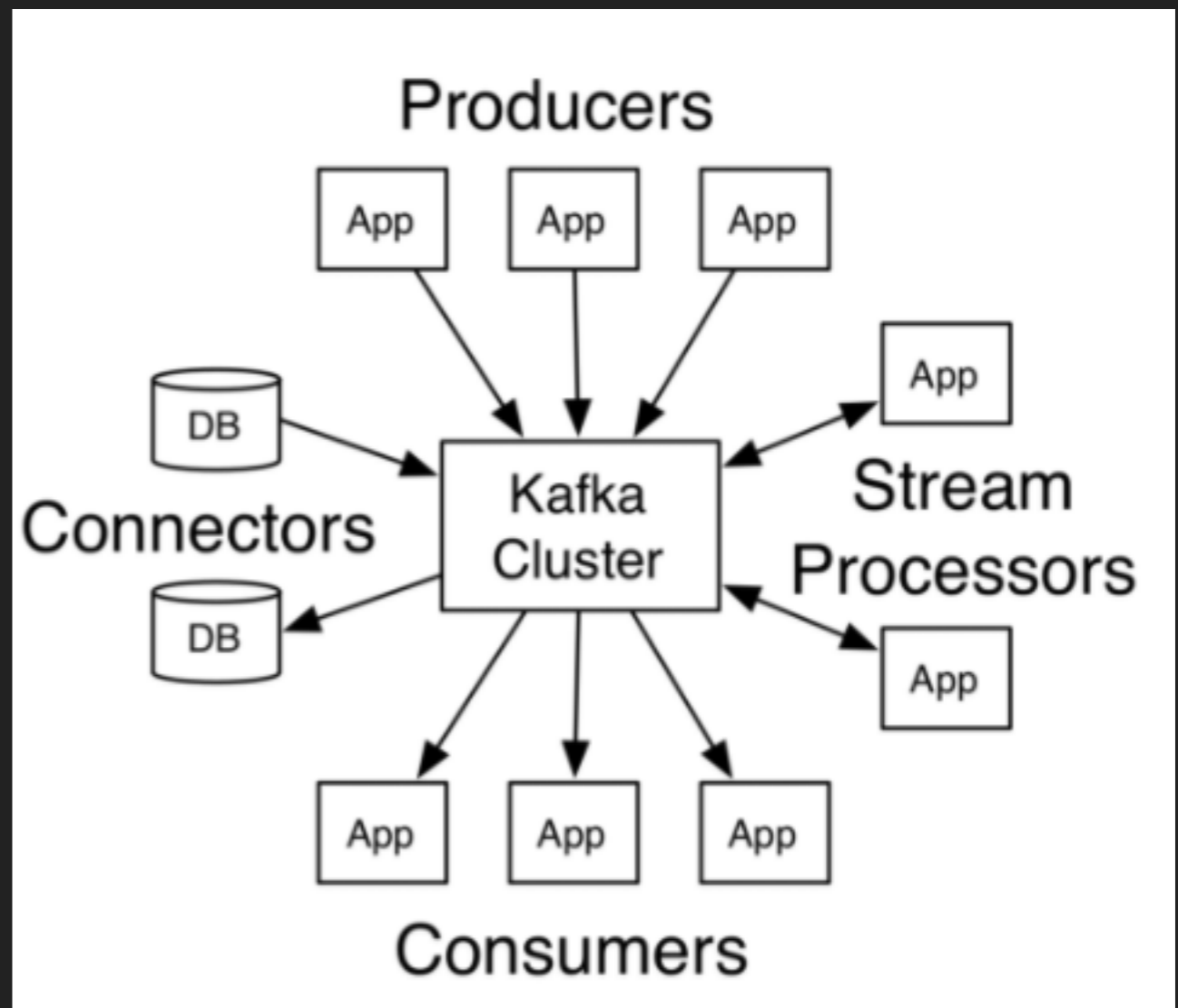
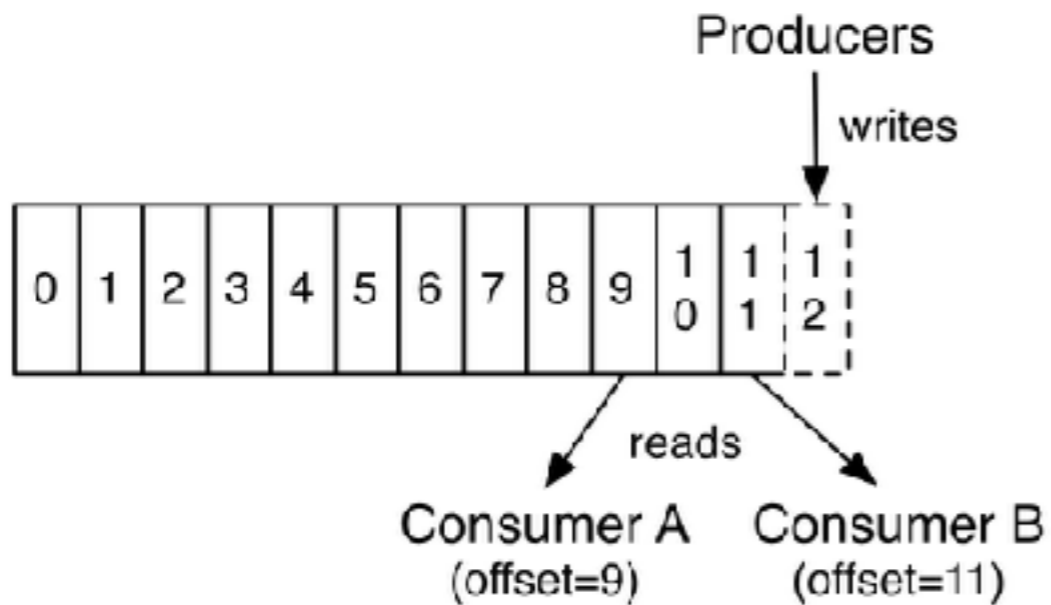
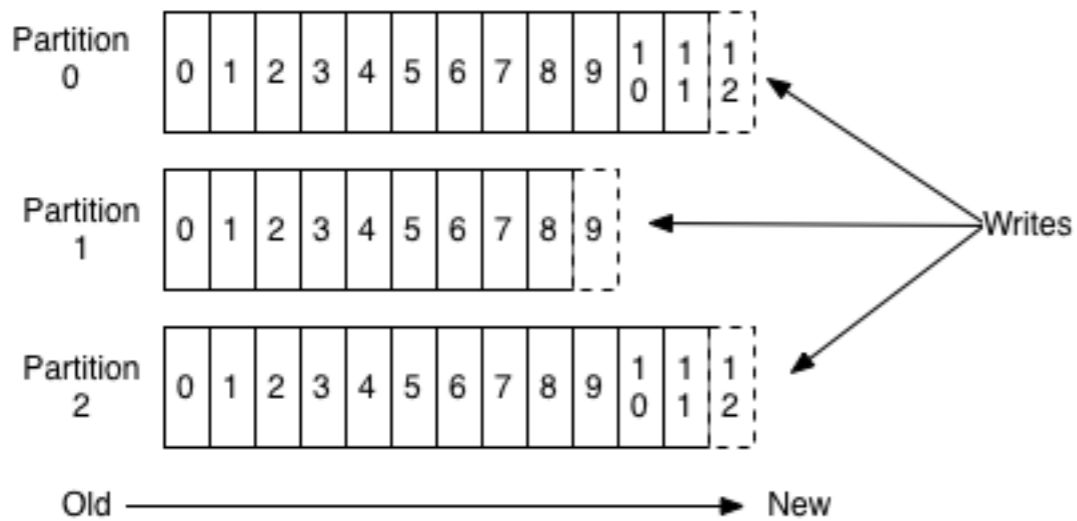
```
1. func (t *RPC) Execute(args *Args, reply *string) {
2.     toolPath := "/usr/bin/exiftool"
3.     // Setup the shell command to launch ExifTool
4.     opts := []string{"-"}
5.     cmd := exec.Command(toolPath, opts)
6.     cmd.Stdin = bytes.NewReader(args.Data)
7.     var out bytes.Buffer
8.     cmd.Stdout = &out
9.     err := cmd.Run()
10.    // - debug output fmt.Println(out.String())
11.    *reply = out.String()
12. }
```

SCARF: DISTRIBUTED EXTENSIBLE ACCOUNTABLE RETRIEVABLE

- ▶ 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

KAFKA AND MESSAGE LOGS

Anatomy of a Topic



SCARF: DISTRIBUTED EXTENSIBLE ACCOUNTABLE RETRIEVABLE

- ▶ 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

```
2
3 PUT /scarf/file/85000
4 {
5   "id": 85000,
6   "filenames" : ["Lord-of-the-Flies.pdf"],
7   "Createtime": 1501875401,
8   "Modifytime": 1501875401,
9   "Accesstime": 1501875401,
10  "Emodifytime": 1501875401,
11  "Fflags": "",
12  "Flags": "",
13  "Filesize": 285166,
14  "Dataruns": []
15 }
```

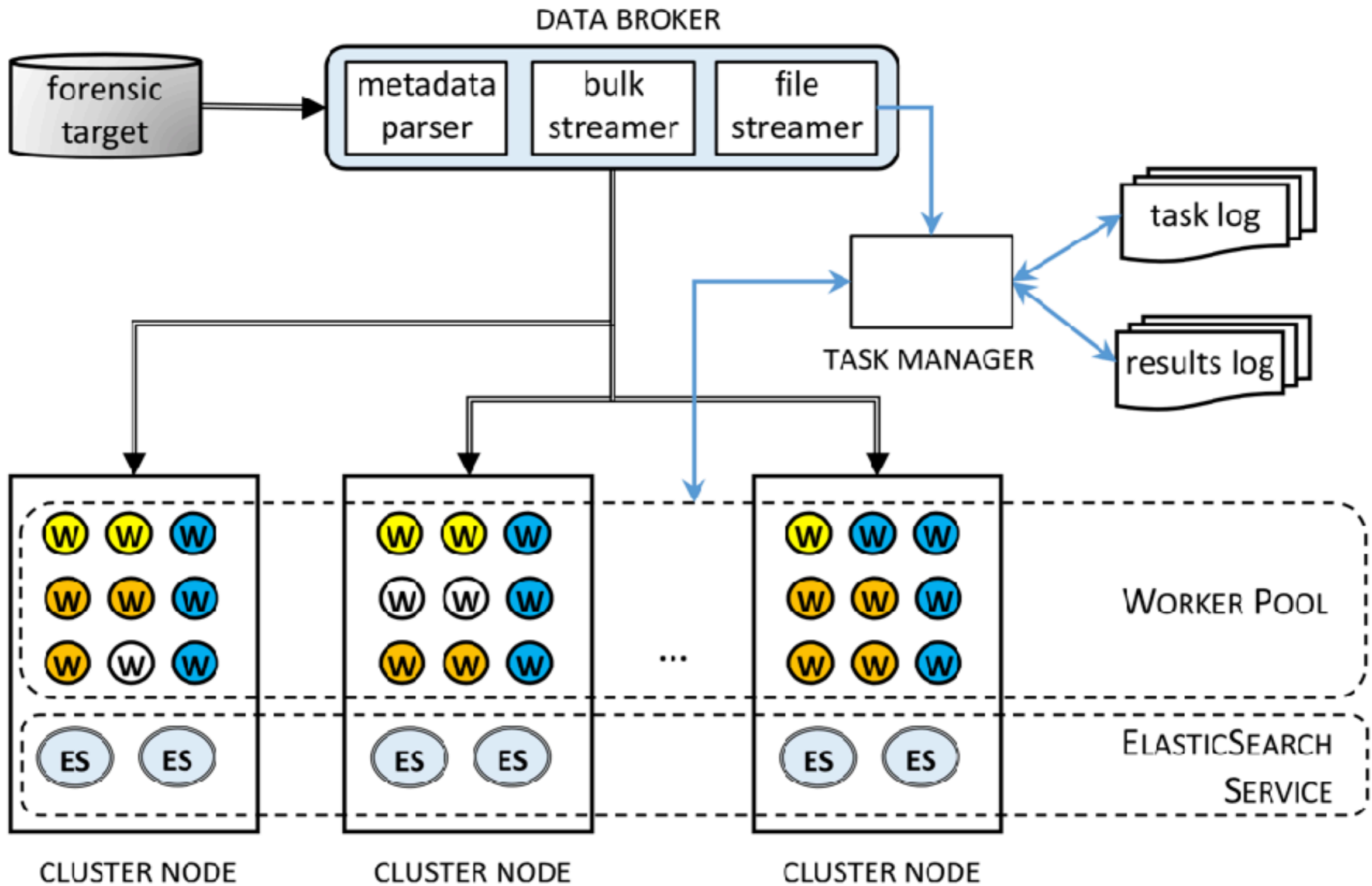
SCARF: DESIGN

- ▶ SCARF is a *framework* built to support digital forensics
 - ▶ **Distributed** - Docker Swarm
 - ▶ **Extensible** - Docker Container + RPC method
 - ▶ **Accountable** - Apache Kafka
 - ▶ **Retrievable** - ElasticSearch
- ▶ Open Source

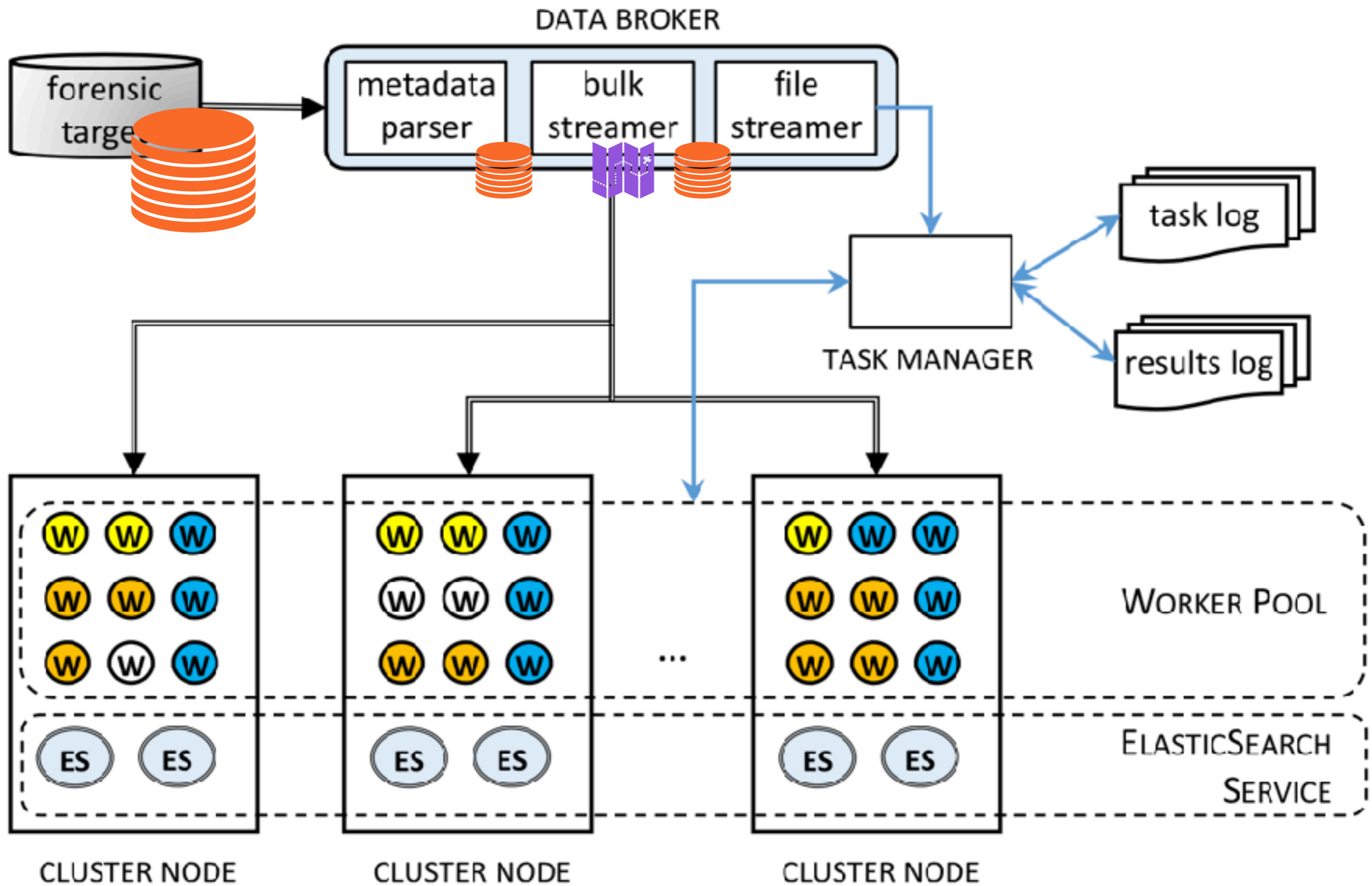
ARCHITECTURE - HOW DO THE PIECES FIT TOGETHER?

- ▶ 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

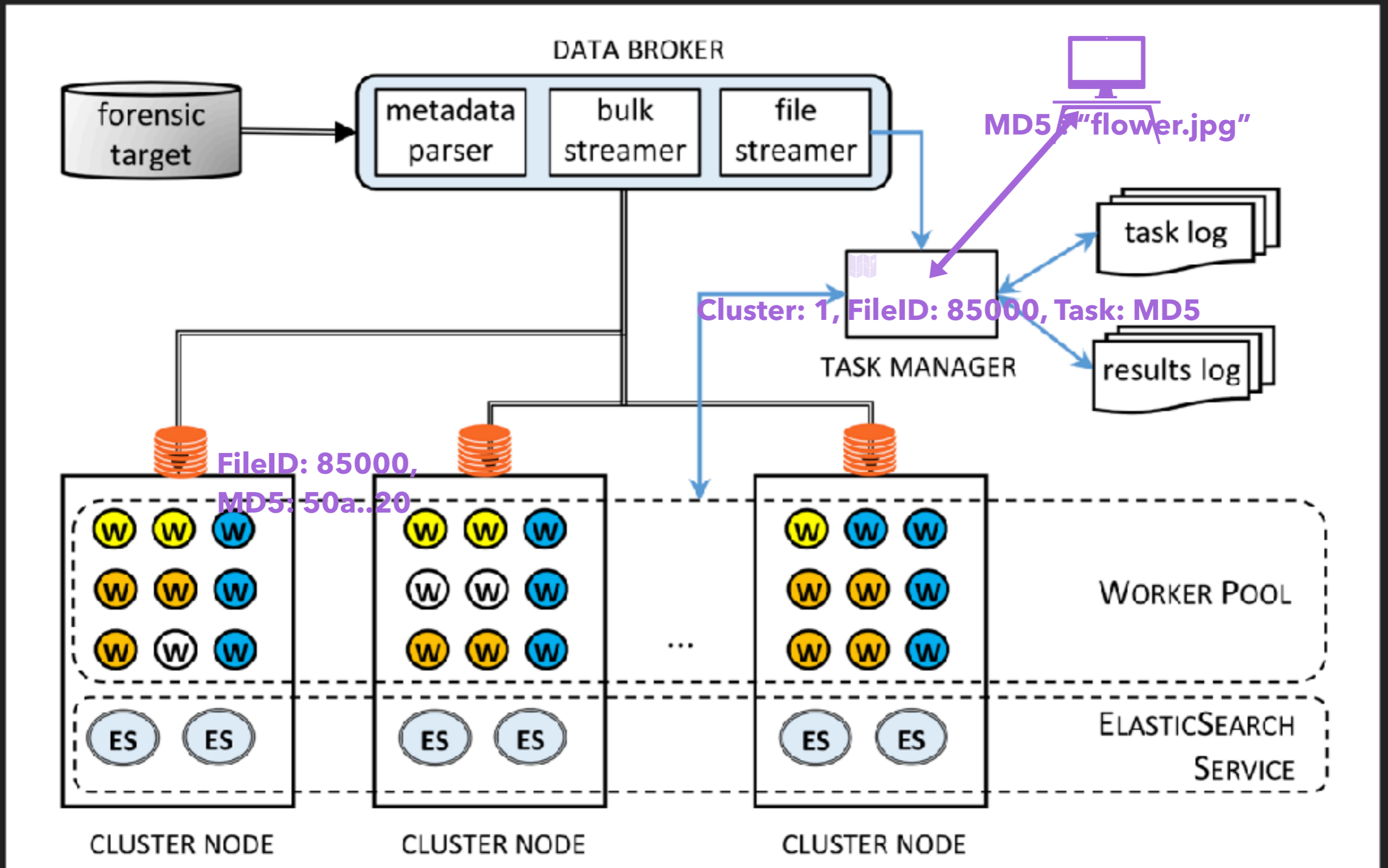
SCARF: SCALABLE REALTIME FORENSICS



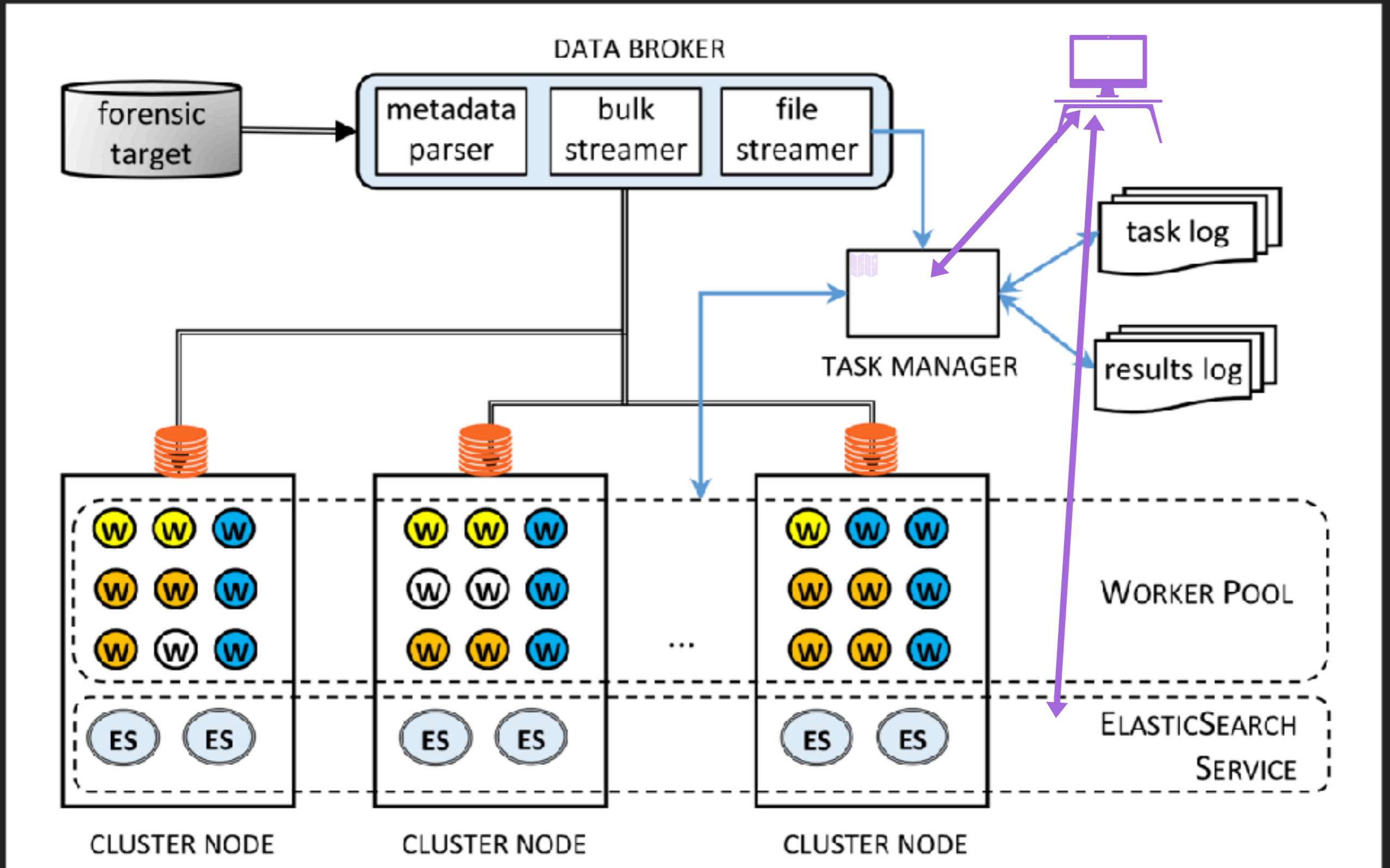
SCARF: SCALABLE REALTIME FORENSICS



SCARF: SCALABLE REALTIME FORENSICS



SCARF: SCALABLE 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

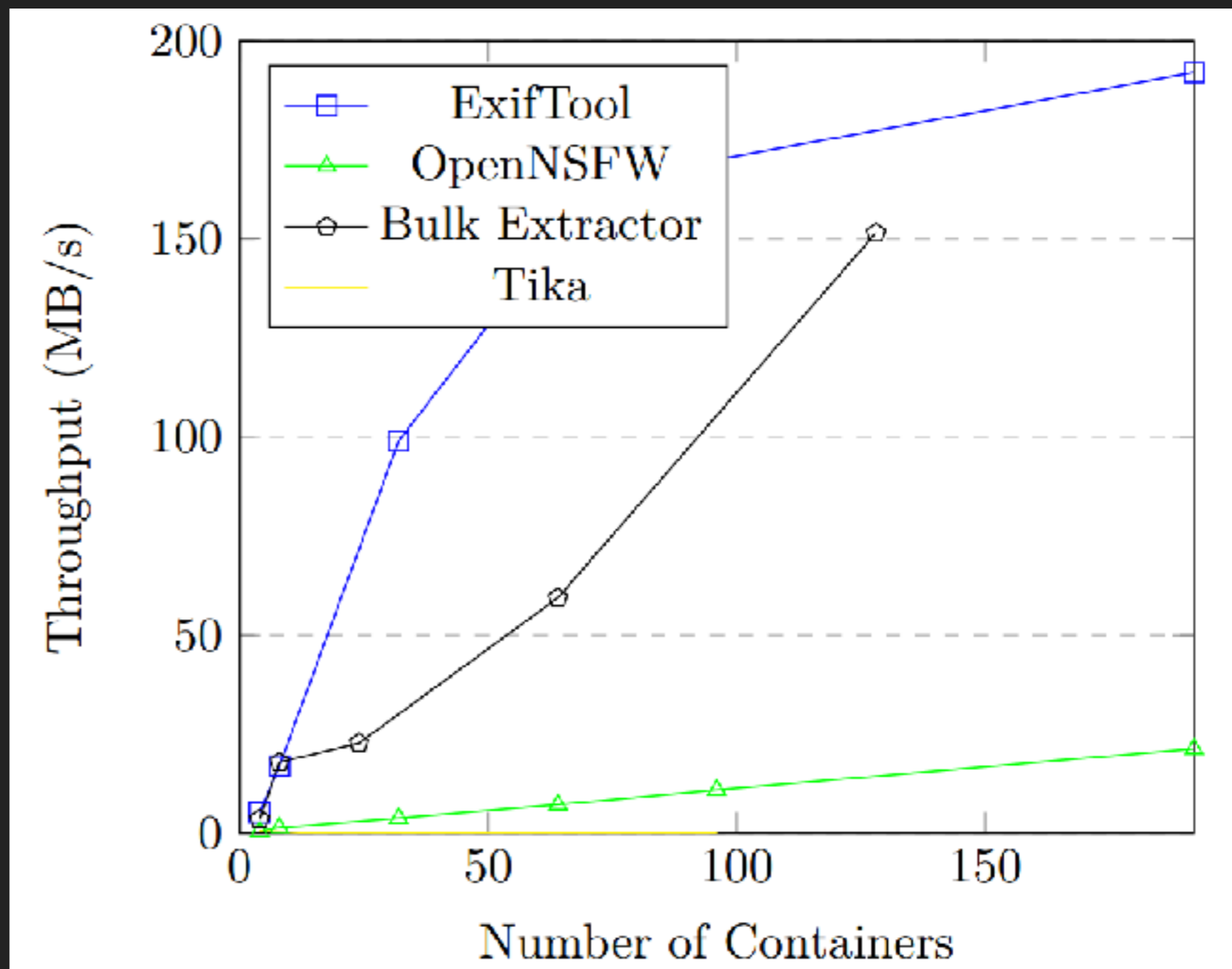
▶ Older Server:

4 nodes

24 cores each

256GB RAM each

▶ 96 total CPUs



NUMBERS

SHA1 # containers	4	8	32	48	96	192
MB/s	345	857	924	985	948	992

OpenNSFW # containers	4	8	32	64	96	192
MB/s	0.4	1.4	3.8	7.2	10.9	21.3

ExifTools # containers	4	8	32	64	96	192
MB/s	5.2	17	99	151	170	192
Per container	1.3	2.1	3.1	2.4	1.8	1

NUMBERS

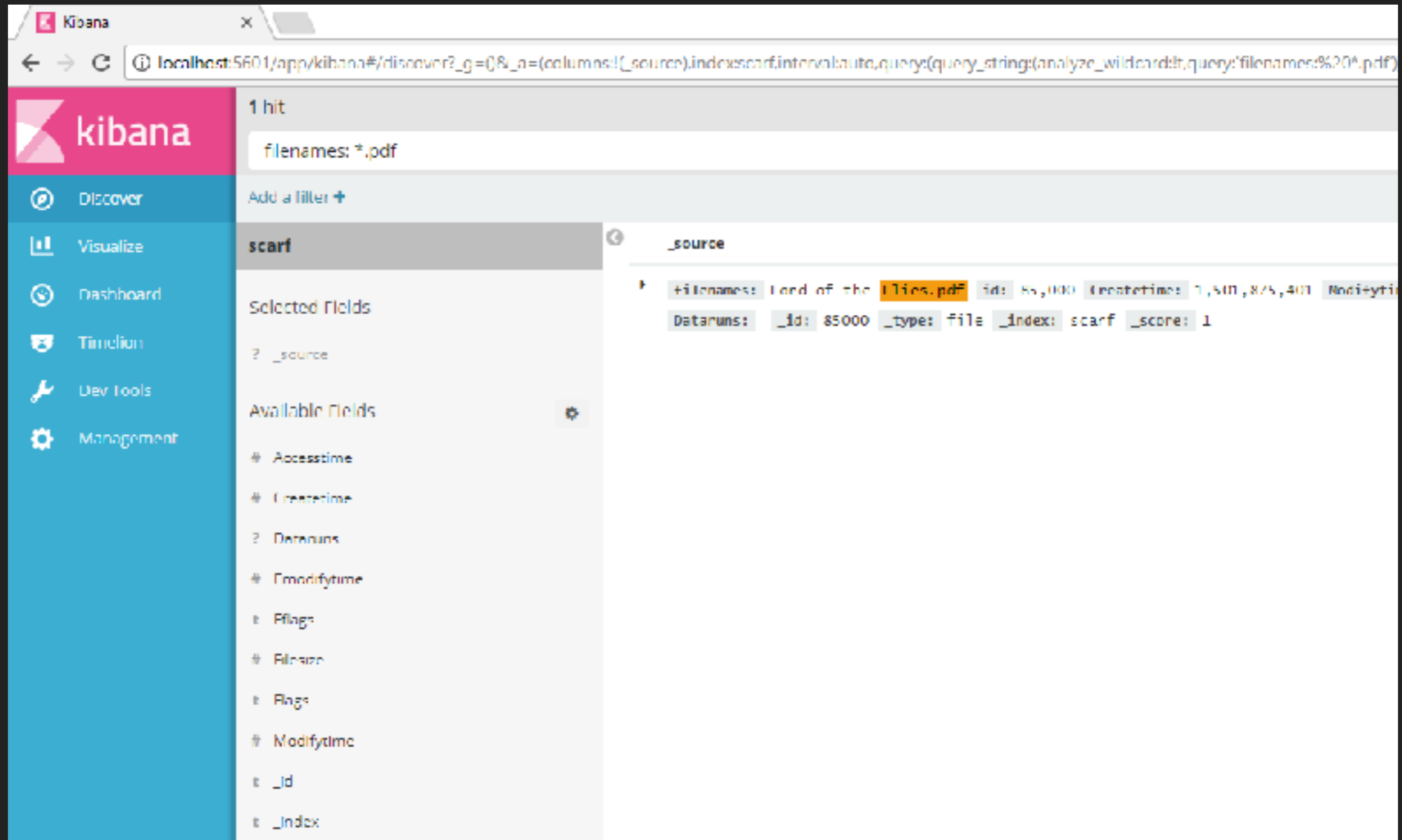
Tika # containers	4	12	24	48	96	192
MB/s	0.5	1.1	2.4	3.5	5.8	6.7
Per container	0.13	0.09	0.10	0.07	0.06	0.03

BulkExtractor # containers	4	12	24	48	64	128
MB/s	3.5	17.9	22.7	54.8	59.4	151.5
Per container	0.9	1.5	0.9	1.1	0.9	1.2

SCREENSHOTS

```
2
3 PUT /scarf/file/85000
4 {
5   "id": 85000,
6   "filenames" : ["Lord-of-the-Flies.pdf"],
7   "Createtime": 1501875401,
8   "Modifytime": 1501875401,
9   "Accesstime": 1501875401,
10  "Emodifytime": 1501875401,
11  "Fflags": "",
12  "Flags": "",
13  "Filesize": 285166,
14  "Dataruns": []
15 }
```

SCREENSHOTS



Searching for all PDFs

SCREENSHOTS

The screenshot shows the Kibana interface with a sidebar on the left containing navigation options: Discover, Visualize, Dashboard, Graph, Monitoring, Timeline, Management, and Dev Tools. The main content area displays the document ID 'scarf/file/1255' and a table view of EXIFTool results. The table has columns for field names and their corresponding values.

Field Name	Value
ExifTool Version Number	: 9.46
File Type	: Win32 EXE
MIME Type	: application/octet-stream
Machine Type	: Intel 386 or later, and compatibles
Time Stamp	: 2001:08:17 20:55:48+00:00
PE Type	: PE32
Linker Version	: 7.0
Code Size	: 19456
Initialized Data Size	: 7168
Uninitialized Data Size	: 0
Entry Point	: 0x1e0f
OS Version	: 5.1
Image Version	: 5.1
Subsystem Version	: 4.0
Subsystem	: Windows command line
File Version Number	: 5.1.2600.0
Product Version Number	: 5.1.2600.0
File Flags Mask	: 0x000f
File Flags	: (none)
File OS	: Windows NT 32-bit
object File Type	: Executable application
File Subtype	: 0
Language Code	: English (U.S.)
Character Set	: Unicode
Company Name	: Microsoft Corporation
File Description	: Router Console Monitor
File Version	: 5.1.2600.0 (xpclient.010817-1146)
Internal Name	: routemon.exe
Legal Copyright	: © Microsoft Corporation. All rights reserved.
original filename	: routemon.exe
Product Name	: Microsoft® Windows® Operating System
Product Version	: 5.1.2600.0

EXIFTool Results

The screenshot shows the Kibana interface displaying the document ID 'scarf/file/5945' and a table view of OpenNSFW results. The table lists various metadata fields and their values.

Field Name	Value
Accesstime	November 20th 2009, 11:49:17.000
Createtime	November 20th 2009, 11:49:17.000
Dataruns	-
Emodifytime	December 31st 0000, 18:00:00.000
Fflags	
Filenames	prague.jpg
Filesize	38,850
Flags	
Id	5,945
Modifytime	April 14th 2008, 07:00:00.000
_id	5945
_index	scarf
_score	1
_type	file
data	{"NSFW-Score": "NSFW score: 0.000425534963142\n"}
fileid	5,945
jobtype	opennsfw
result	

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)

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?