DFRWS'12 / Aug 5-8 2012 / Washington DC

Content triage with similarity digests: The M57 case study

THE UNIVERSITY of NEW ORLEANS

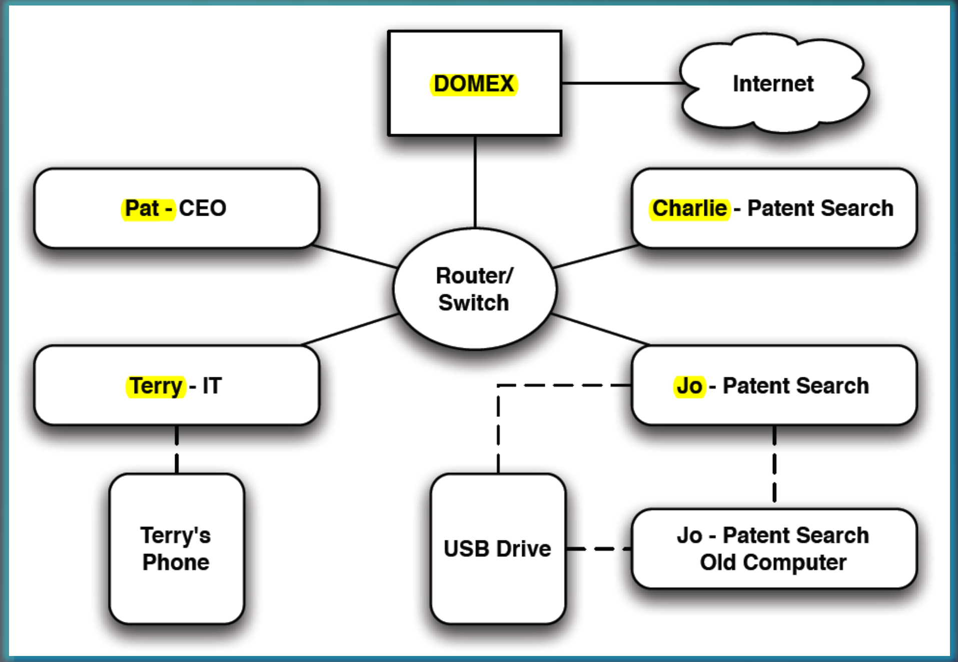
Vassil Roussev

Candice Quates

The M57 Case Study

Introduction

M57: The company & setup



- Employees:
  - President: Pat McGoo
  - IT: Terry
  - Researchers: Jo, Charlie
- Period
  - 11/16/2009—12/11/2009
  - 11/20/2009 Jo's computer replaced
  - Last day: police kick down the door
- Data
  - Daily HDD, RAM, network captures

M57: The data (1.5 TB)

- HDD images
  - 84 images, 10-40GB each
  - Total: **1,423 GB**
- RAM snapshots
  - 78 snapshots, 256-1024 MB each
  - Total: **107 GB**
- Network:
  - 49 traces, 4.6 GB
- USB
  - 4.1 GB
- Kitty set
  - 125 JPEGs, 224 MB

Scenario #1: Contraband

- Setup:
  - From the detective reports in the scenario, there is reason to suspect that one of M57's computers (Jo’s) has been used in the contraband of "kitty porn".
- Questions:
  - Were any M57 computers used in contraband?
  - If so, when did the accident happen?
  - Is there evidence of intent?
  - How was the content distributed?
  - Was any of the content sent outside the company network?

Scenario #2: Eavesdropping

- Setup:
  - It is suspected that somebody is spying on the CEO (Pat) electronically.
- Plan?
  - Search for potentially rogue processes that might have been introduced on his computer.
  - First HDD image is clean and serves as baseline.

Scenario #3: Corporate espionage

- Setup:
  - There is suspicion that somebody has leaked company secrets.
- Plan?
  - Search RAM snapshots for interesting processes

The need for better triage

Triage

- ***Fast, reliable*** initial screen of the acquired data:
  - *fast:* all you can do in 5/10/15/ … min;
  - *reliable:* provides *strong hints* (low FP).
- Goals:
  - Identify the most (ir)relevant targets/artifacts;
  - Build an overall understanding of the case—what are the likely answers?
- Location of work:
  - We assume post-acquisition work in a lab, but
  - It could be done in the field (given enough hardware)

Metadata- vs content-based analysis

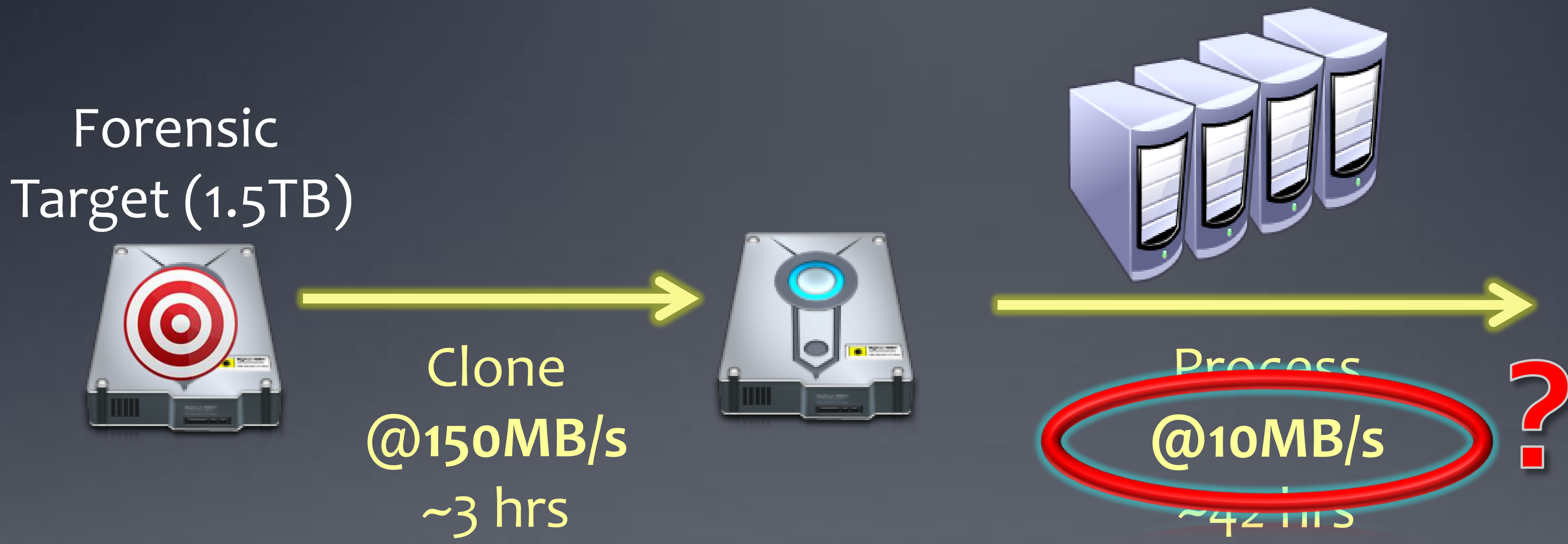
- Metadata-based analysis
  - Use FS metadata, registry, logs, etc.
  - Pro: small volume, high-level logical information
  - Con: not looking at the data, cannot see remnants, does not work on a data dump (e.g. RAM), metadata is fragile

  ➔ Typical basis for (manual) triage

- Content analysis
  - Works on actual data content
    - Flie/block hashes, indexing, carving, etc.
  - Pro: looking at actual data, can work with pieces
  - Con: large volume, lower level data

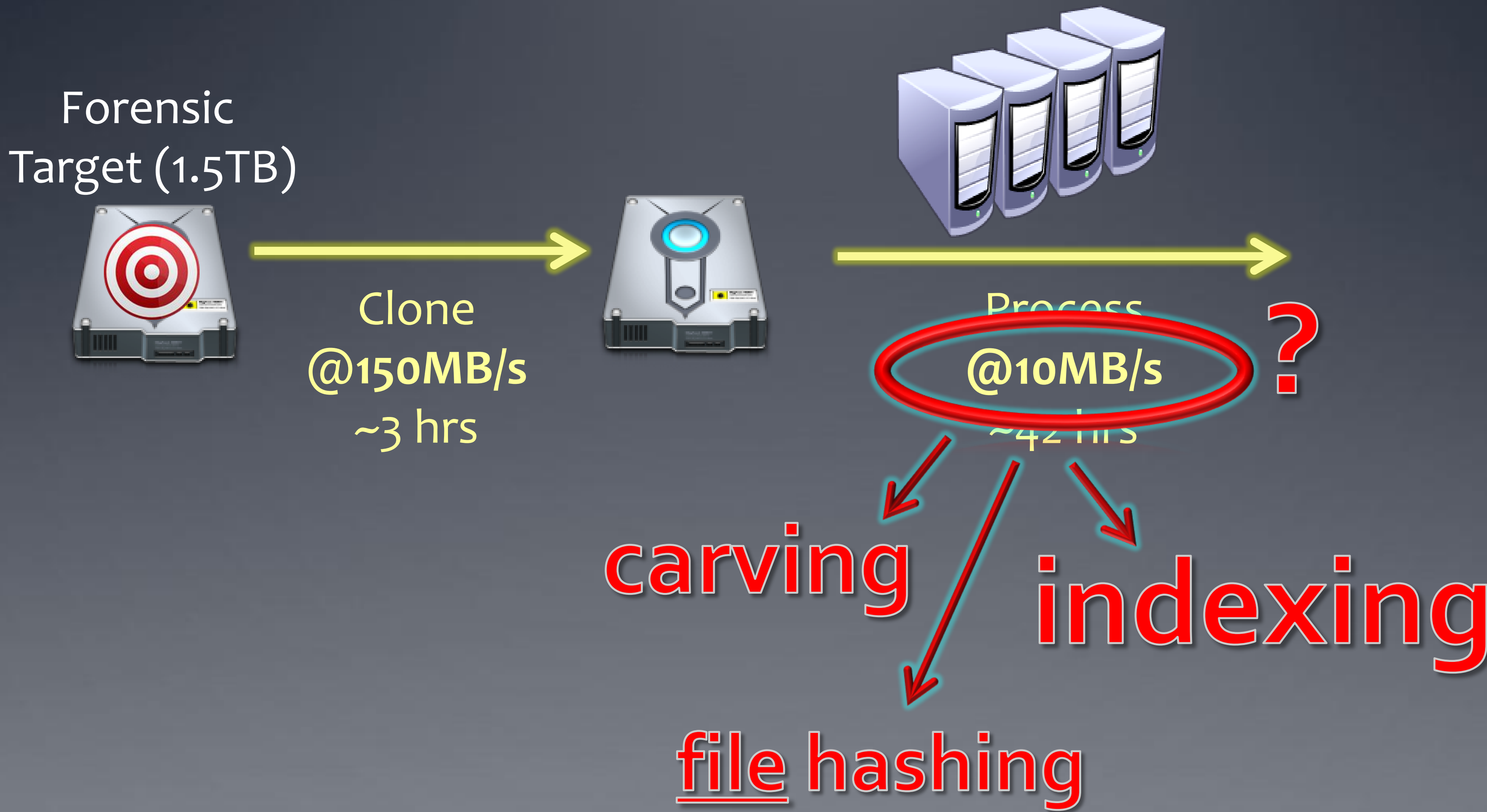
  ➔ Almost never used in triage (perceived as too slow)

Why is content analysis so slow?

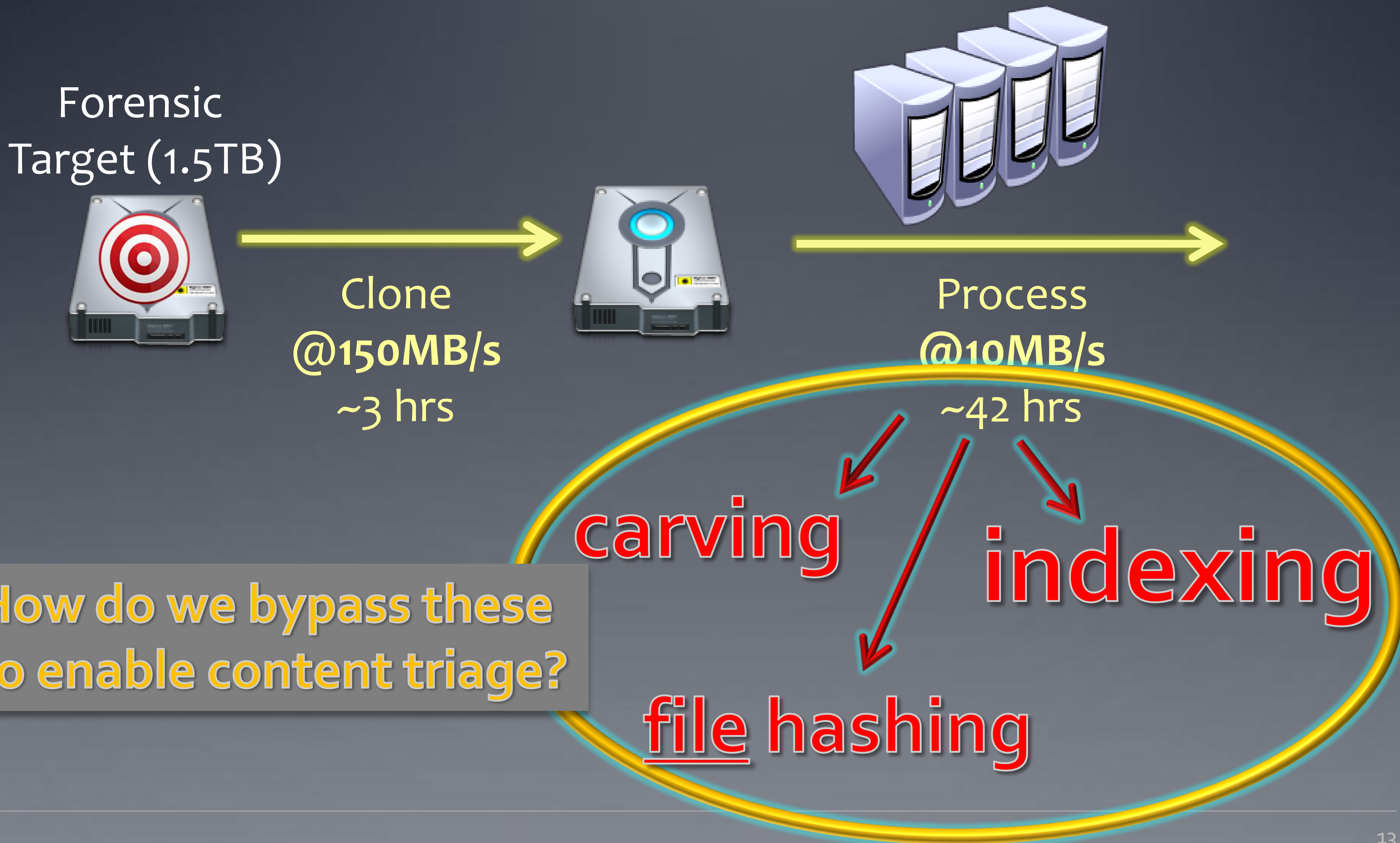


→ We can *start* working on the case after 42 hours (!)

Why is content analysis so slow?



Why is content analysis so slow?



How do we bypass these to enable content triage?

Data Correlation

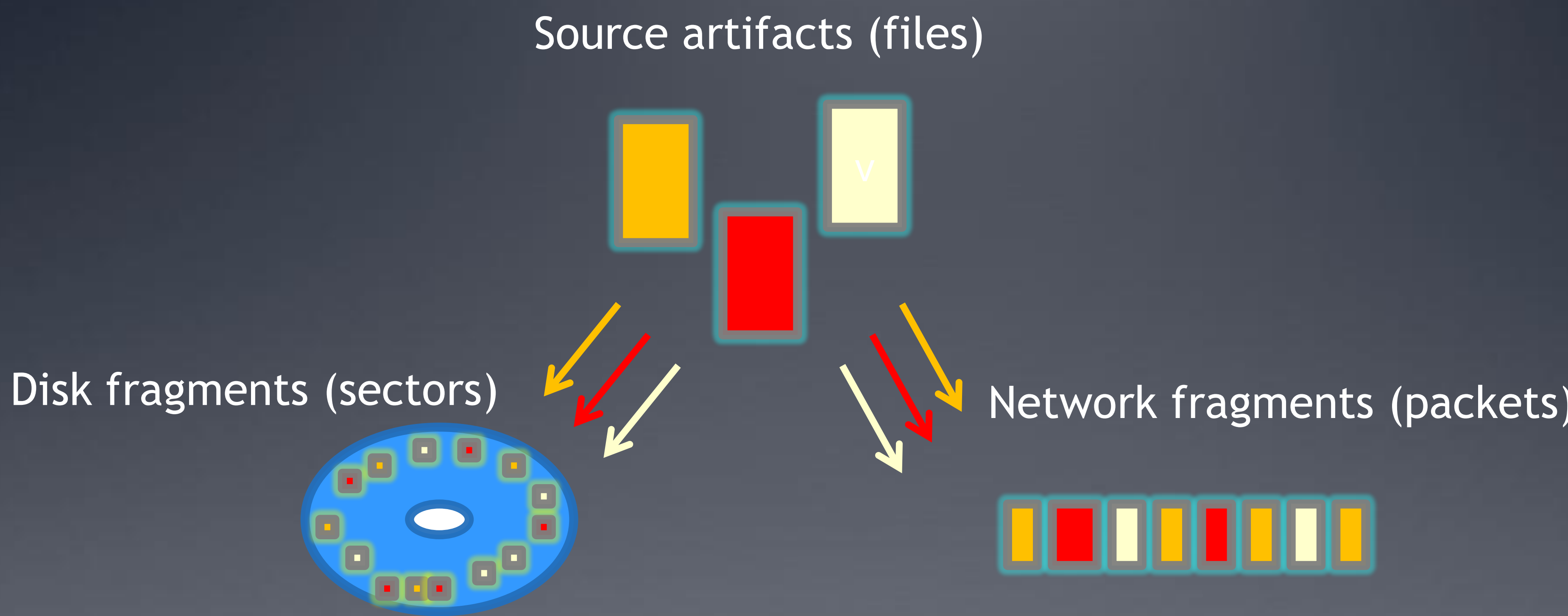
with similarity digests

Motivation for similarity approach: *Traditional hash filtering is failing*

- *Known file filtering:*
  - Crypto-hash known files, store in library (e.g. NSRL)
  - Hash files on target
  - Filter in/out depending on interest
- Challenges
  - Static libraries are falling behind
    - Dynamic software updates, trivial artifact transformations

    ➔ We need **version** correlation
  - Need to find embedded objects
    - Block/file in file/volume/network trace
  - Need higher-level correlations
    - Disk-to-RAM
    - Disk-to-network

Scenario #1: fragment identification



- Given a fragment, identify source
  - ***Minimum*** fragments of interest are 1-4KB in size
  - Fragment ***alignment is arbitrary***

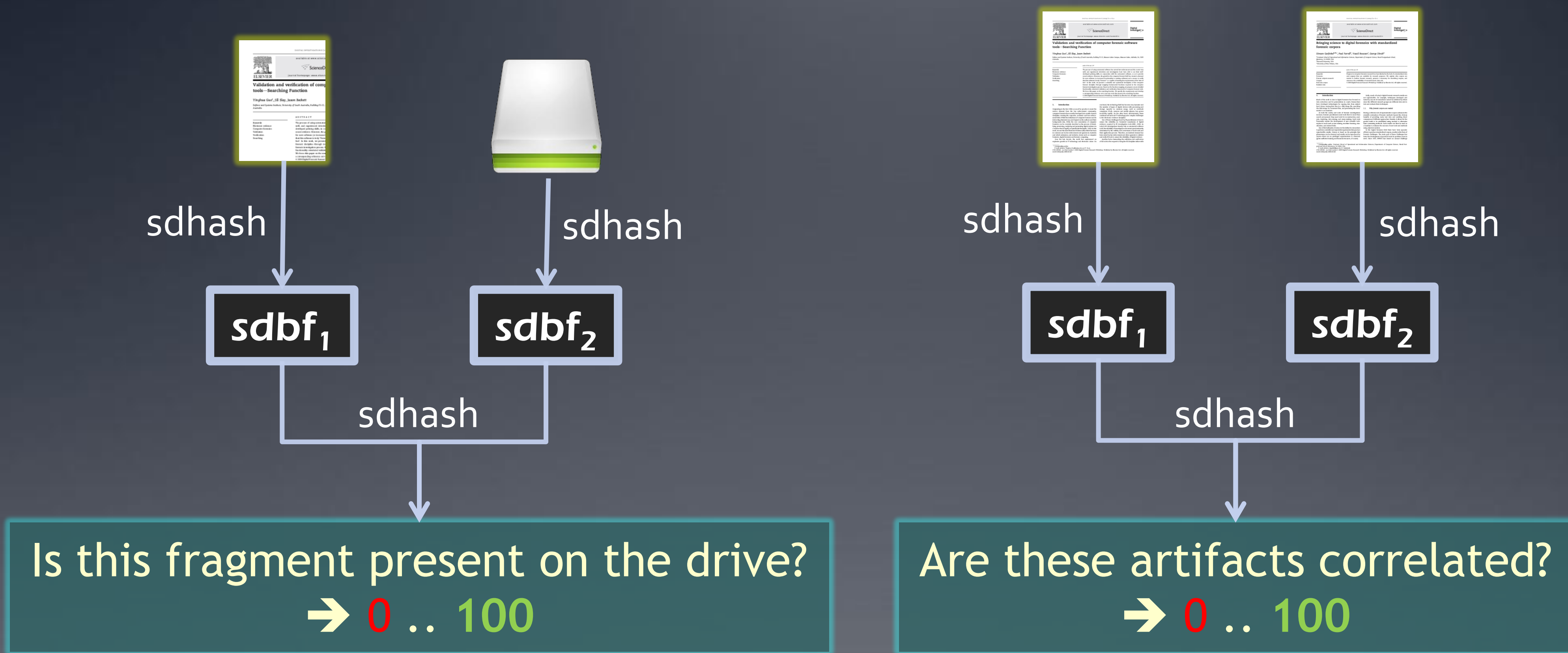
Scenario #2: artifact similarity

Similar files
(shared content/format)

Similar drives
(shared blocks/files)

- Given two binary objects, detect similarity/versioning
  - Similarity here is purely syntactic;
  - Relies on commonality of the binary representations.

Common solution: similarity digests

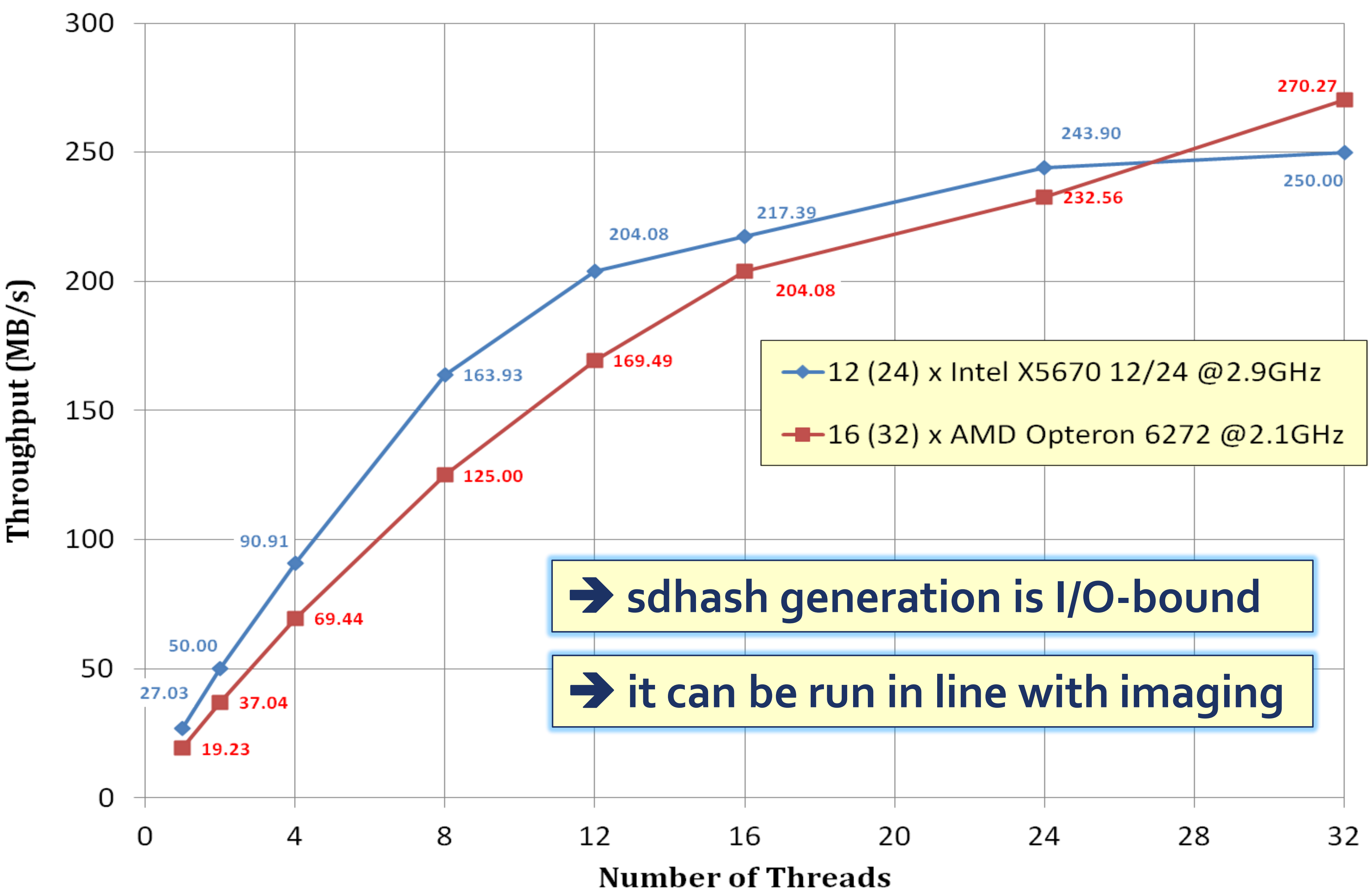


All correlations based on bitstream commonality

The M57 Case Study

Using sdhash for triage

sdhash-2.2 generation rates



➔ sdhash generation is I/O-bound

➔ it can be run in line with imaging

sdhash generation times (M57)

Data Set	Size (GB)	Time (min)	Rate (MB/s)
HDD	1,423.0	168.00	143
RAM	107.0	10.70	166
Network	4.6	0.40	196
USB disk	4.1	0.45	155
Kitty	0.2	0.08	45
Total	**1,538.9**	**179.63**	**143**

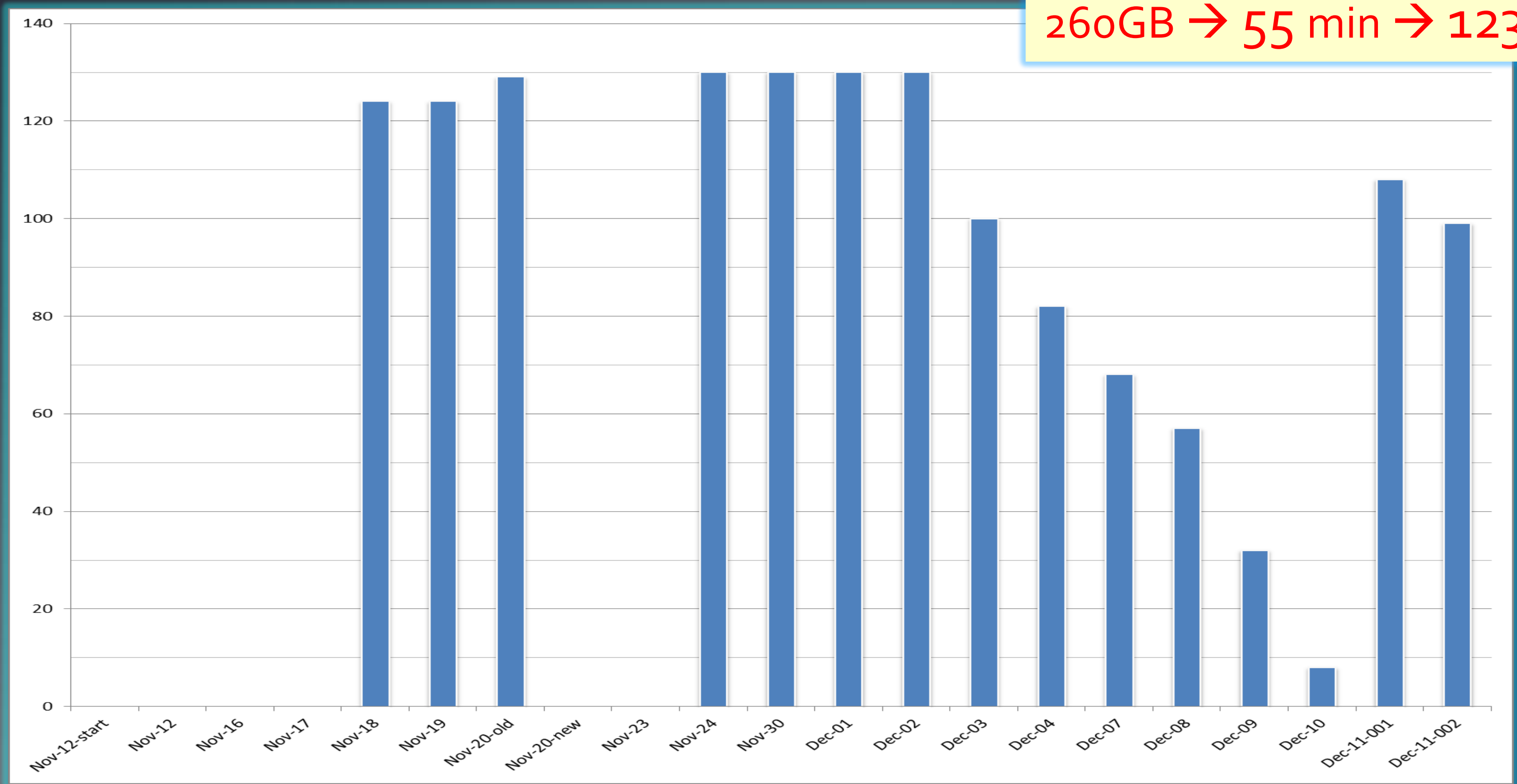
- Dell PowerEdge R710 server
  - 2 x Intel Xeon CPUs @2.93GHz six-core with H/T 12(24) threads
  - 72GiB of RAM @800MHz

Scenario #1: Contraband

- Setup:
  - From the detective reports in the scenario, there is reason to suspect that one of M57's computers (Jo’s) has been used in the contraband of "kitty porn".
- Questions:
  - Were any M57 computers used in contraband?
  - If so, when did the accident happen?
  - Is there evidence of intent?
  - How was the content distributed?
  - Was any of the content sent outside the company network?

Query 1: Search Jo's HDD for kitty images

260GB → 55 min → 123 sec



Jo's computer: Number of instances found by date

Query 2: What processes were running?

- Search Jo's RAM for traces of installed executables

18 min

```
12/03
.../Downloads/TrueCrypt Setup 6.3a.exe 092
.../TrueCrypt Format.exe               090
.../TrueCrypt Setup.exe                092
.../TrueCrypt.exe                      092
12/04
.../Downloads/TrueCrypt Setup 6.3a.exe 063
.../TrueCrypt Setup.exe                063
12/09
.../Downloads/TrueCrypt Setup 6.3a.exe 084
.../TrueCrypt Format.exe               079
.../TrueCrypt Setup.exe                084
.../TrueCrypt.exe                      090
12/10
.../TrueCrypt.exe                      092
12/11 - pre-raid
.../TrueCrypt Format.exe               086
.../TrueCrypt.exe                      079
```

Scenario #2: Eavesdropping

- Setup:
  - It is suspected that somebody is spying on the CEO (Pat) electronically.
- Plan?
  - Search for potentially rogue processes that might have been introduced on his computer.
  - First HDD image is clean and serves as baseline.

Eavesdropping timeline

20 min

```
11/16, [71] not in baseline
  Present: Java, Firefox, python, mdd_1.3.exe.
11/19, [95] not in baseline
  Acrobat Reader 9 installed or updated,
  including Adobe Air.
  18 other programs from 11/16 still present.
11/20, [289]
  Windows Update run: many new dlls in the
  _restore and SoftwareDistribution folders.
11/23, [561]
  Windows Update has run
11/30, [274]
  Likely a Brother printer driver installed.
  Acrobat/Firefox still present.
```

```
12/03, [649]
  AVG has been updated.
  XP Advanced Keylogger appears:
    XP Advanced/DLLs/ToolKeyloggerDLL.dll 087
    XP Advanced/SkinMagic.dll             027
    XP Advanced/ToolKeylogger.exe         024
12/07, [460]
  More Brother printer related files.
  InstallShield leftovers present.
  win32dd present.
  XP Advanced Keylogger is no longer here.
  RealVNC VNC4 has been installed and run:
    RealVNC/VNC4/logmessages.dll 068
    RealVNC/VNC4/winvnc4.exe     046
    RealVNC/VNC4/wm_hooks.dll    023
12/10, [1240]
  AVG updated.
  IE8 and Windows updated.
  VNC still present.
12/11, [634]
  VNC present.
```

Scenario #3: Corporate espionage

- Setup:
  - There is suspicion that somebody has leaked company secrets.
- Plan?
  - Search RAM snapshots for interesting processes

Scenario #3: Findings

31 min

- RAM
  - "Cygnus FREE EDITION" hex editor
    - On 11/24, 11/30, 12/02, 12/03, and 12/10;
  - "Invisible Secrets 2.1“
    - 11/19, 11/20, 11/24, 11/30, and 12/02.
    - `blowfish.dll, jpgcarrier.dll, bmpcarrier.dll`

  ➔ likely stego tool

- USB
  - `insecr2.exe`
  - `/microscope.jpg`
  - `/microscope1.jpg`
  - `/astronaut.jpg`
  - `/astronaut1.jpg`
  - `/Email/Charlie_..._Sent_astronaut1.jpg`
  - `/Email/other/Charlie_..._Sent_microscope1.jpg`

M57 Conclusions

- Using sdhash, we can outline the solution of all three cases in about **120 min** of extra processing.
  - This assumes HDD/RAM hash generation while cloning.
  - This could be further improved by running the queries in R/T in parallel with acquisition.

- The tool enables differential analysis that is simple, fast, robust, and generic.
  - → Most processing can run in parallel with acquisition.
  - → In effect, it can replace carving/indexing during triage.
  - → It does not require much expertise to apply; results are intuitive.
  - → The analysis can be highly automated; higher-level analysis can be built on top.

Development Status

Architecture

C++ Client: `sdhash-cli`

Web GUI (`python`)

Custom clients (20+ languages)

Apache Thrift C/S Protocol

Python

Other

CLI: `sdhash`

Server: `sdhash-srv`

SWIG-based APIs

Cross-platform C++ API: `libsdbf`

Third-party C++ libraries: `boost,thrift,openssl (thrust,TBB)`

Availlability

- sdhash.org
  - Source
  - Windows exe
    - 32-/64-bit executables
  - Linux
    - rpm/deb packages
  - API documentation
  - Repository
  - Papers/presentations

sdhash-2.2 comparison performance

- Small file comparison (1 core, Intel X5670)

```
10KB  vs. 10KB    0.0061 ms
100KB vs. 100KB   0.0125 ms
1MB   vs. 1MB     0.4300 ms
10MB  vs. 10MB   41.0000 ms
```

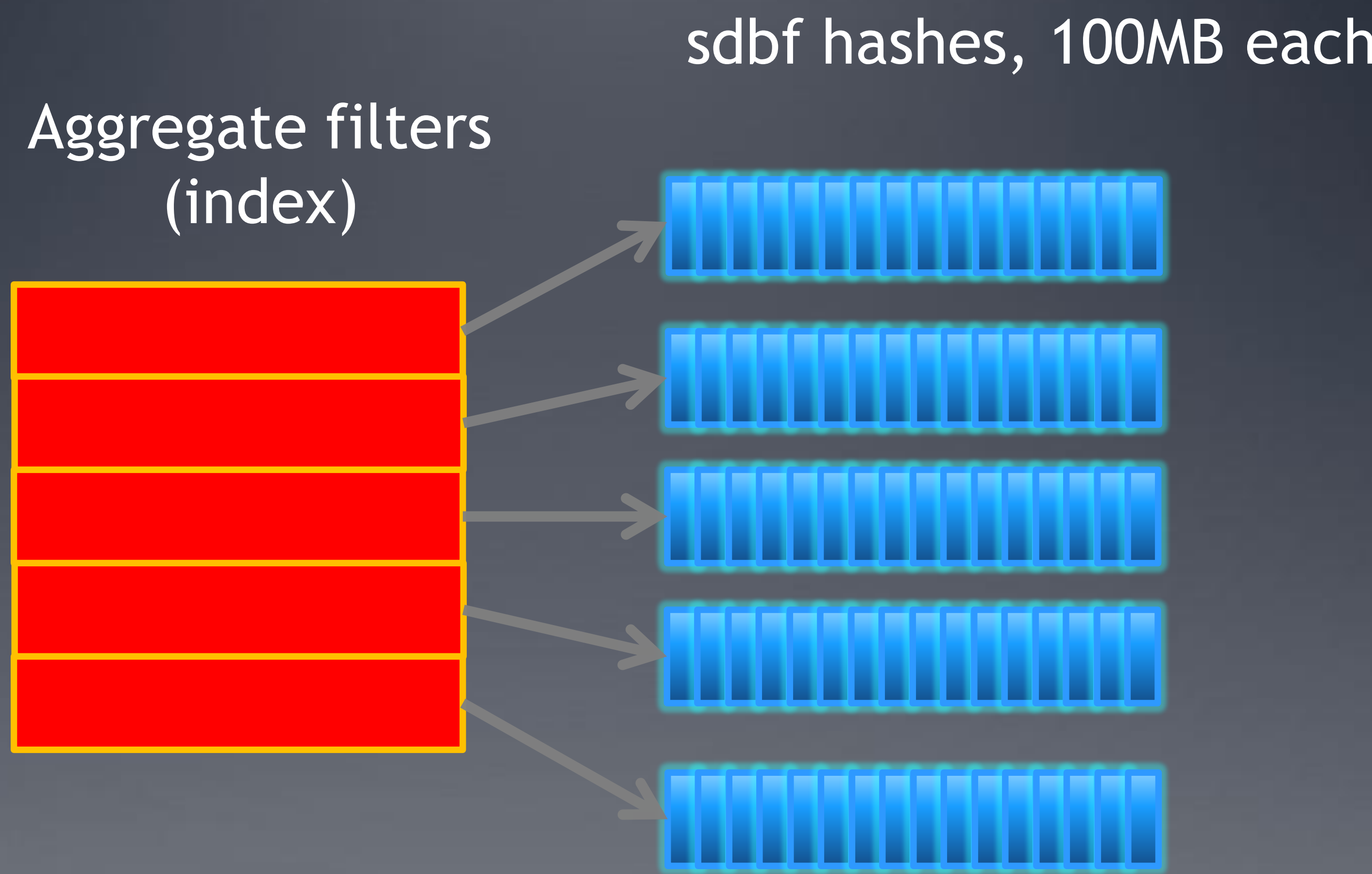
- Large file/streaming comparison (12 cores) in seconds

	100MB	125MB	150MB	200MB	500MB	1000MB
100MB	0.76	0.93	1.00	1.36	3.53	6.61
125MB	0.93	0.96	1.30	1.84	4.10	8.60
150MB	1.00	1.30	1.58	2.28	5.33	10.30
200MB	1.36	1.84	2.28	3.00	7.10	13.80

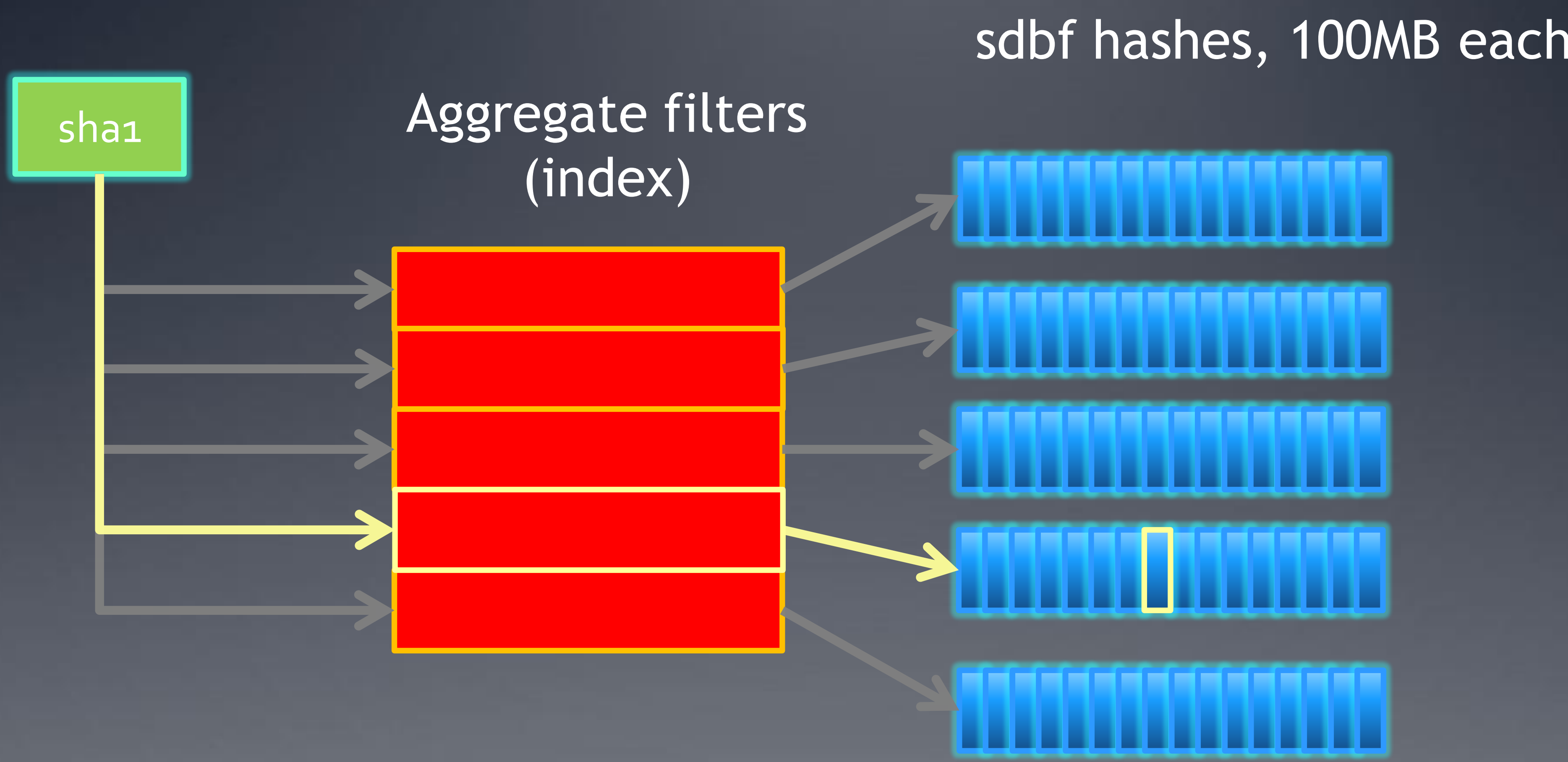
Todo: Scaling up to NSRL

- Goal:
  - Maintain R/T performance (100-150 MB/s) with 1TB reference set.
- Approach:
  - Pre-filtering/indexing using extra Bloom filters
- Estimated cost:
  - Approximately 2.5% extra; i.e. increase from 2.5 to 5% of reference data
  - 50GB per TB of data
  - Requires RAM-optimized server (e.g. 256GB → ~$7k)

Scaling up to NSRL (2)



Scaling up to NSRL (2)



Todo list

- ***libsdbf***
  - Rewrite parallelization using *thrust*, *tbb*, *thrift*, or similar
  - Implement pre-filtering/indexing
  - GPU acceleration
- ***sdhash***
  - More command line options/compatibility w/ssdeep
  - Pcap front end
    - payload extraction, file discovery, time-lining
- ***sdhash-srv/sdhash-cli***
  - Multi-server deployment
  - GUI

Further Development

- Integration w/ RDS
  - ***sdhash-set***: construct *SDBF*s from existing SHA1 sets
    - Compare/identify whole folders, distributions, etc.
- Structural feature selection
  - E.g., exe/dll, pdf, zip, …
- Optimizations
  - Skipping
    - Under **min** continuous block assumption
  - Cluster "core" extraction/comparison
- Representation
  - Multi-resolution digests
  - New crypto hashes
  - Data offsets

Thank you!

- http://sdhash.org

- sdhash tutorial: Wed, Aug 8 @3pm

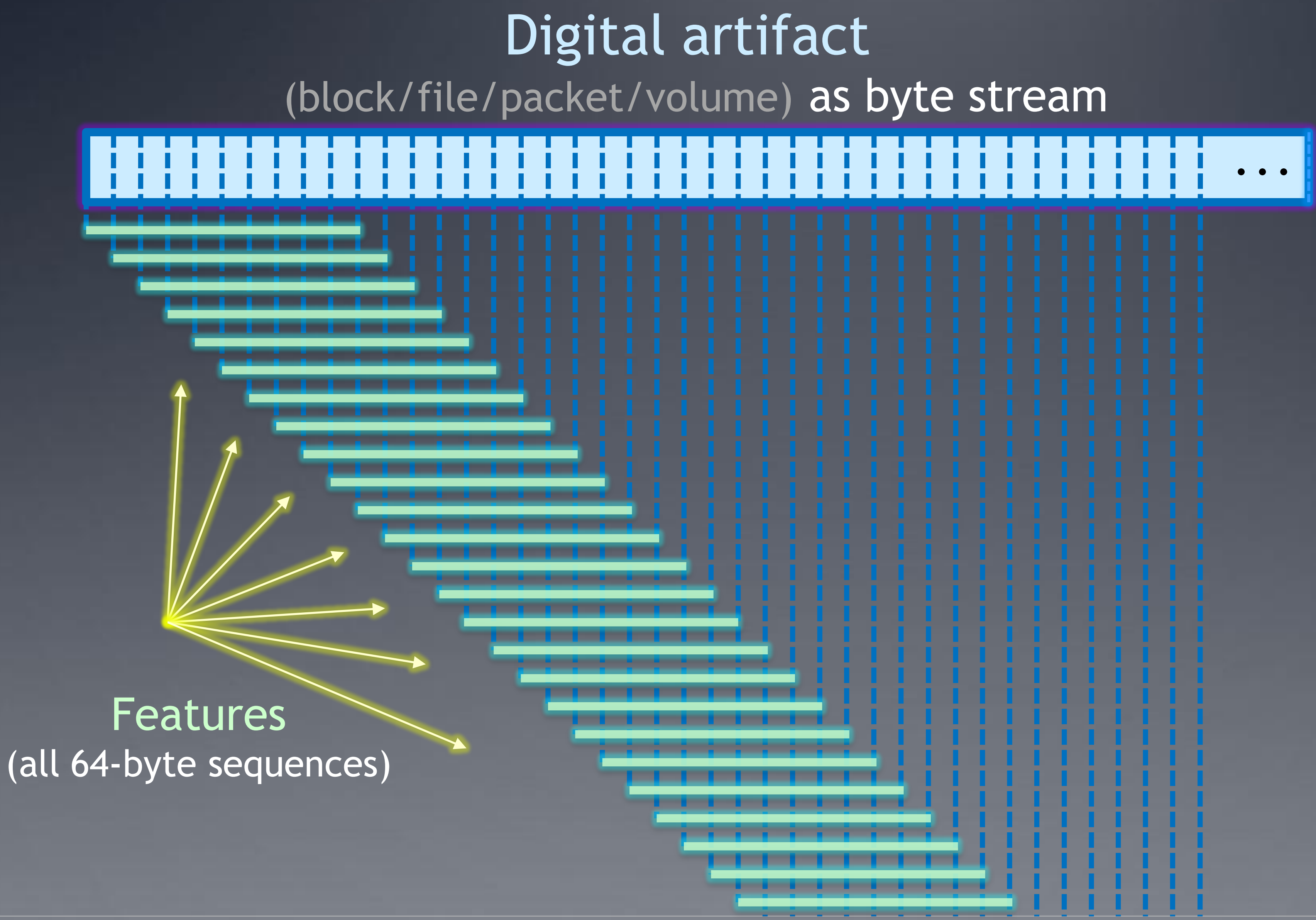
- Vassil Roussev

  vassil@roussev.net

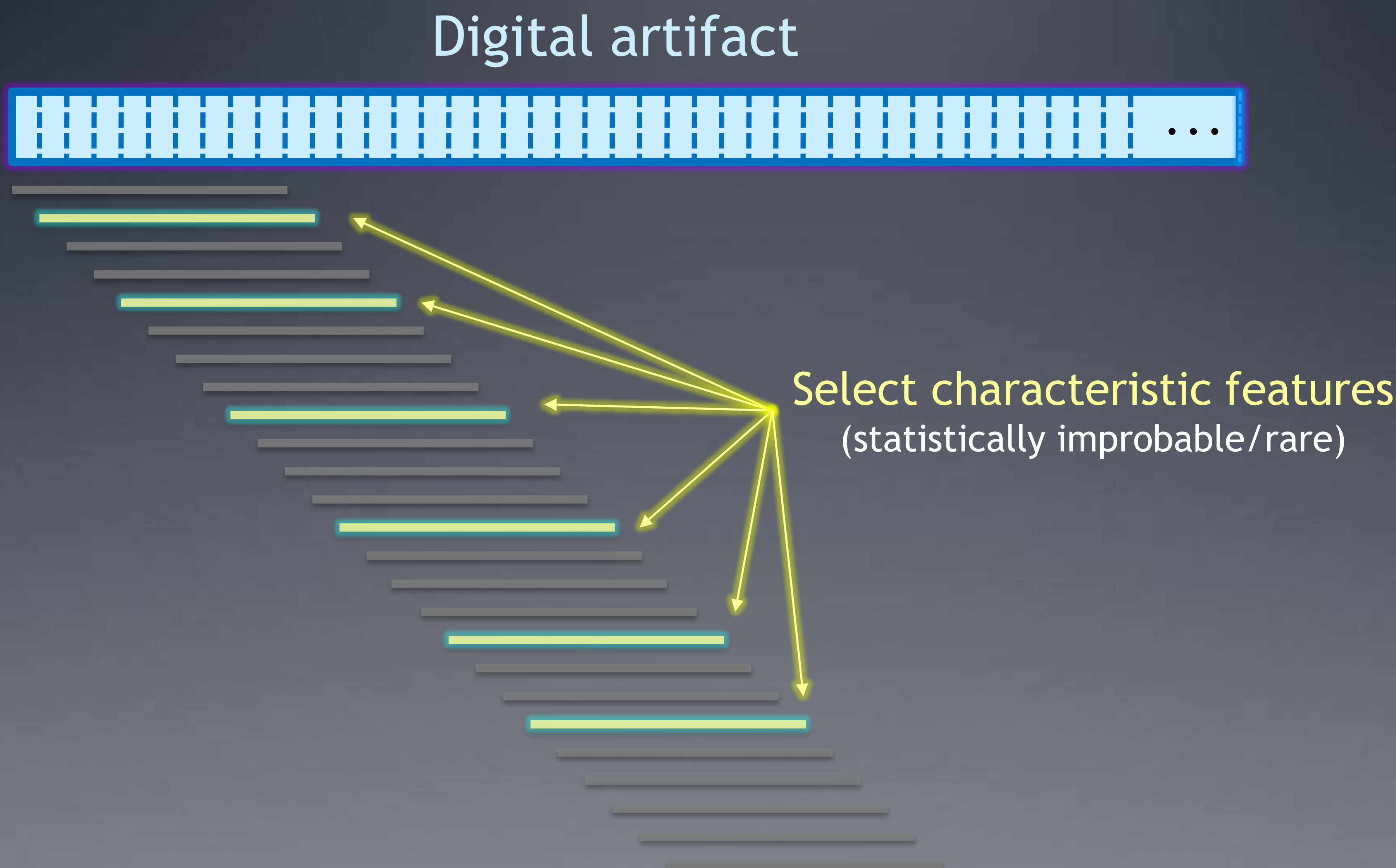
Similarity digests

Overview

Generating sdhash fingerprints (1)

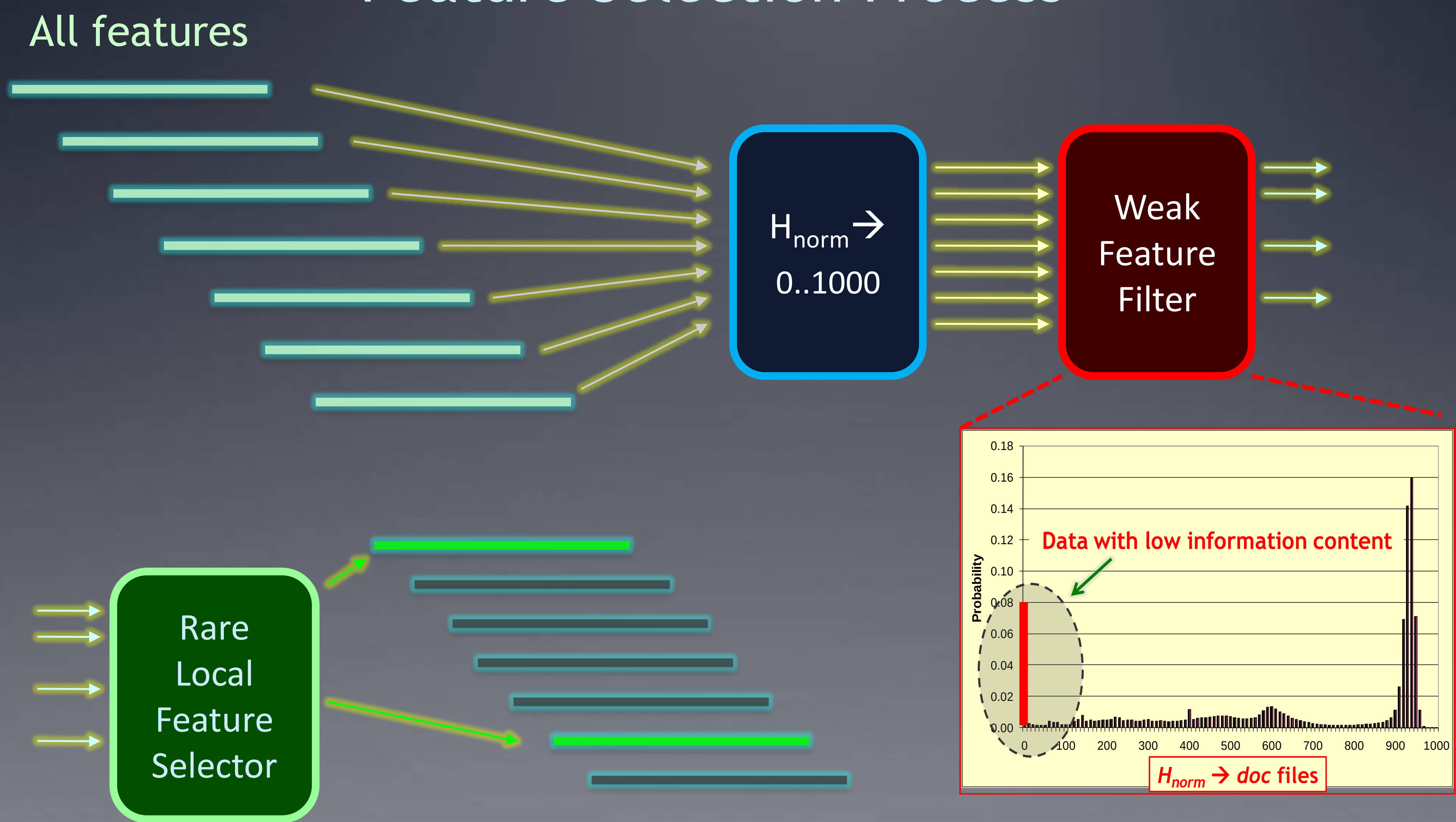


Generating sdhash fingerprints (2)

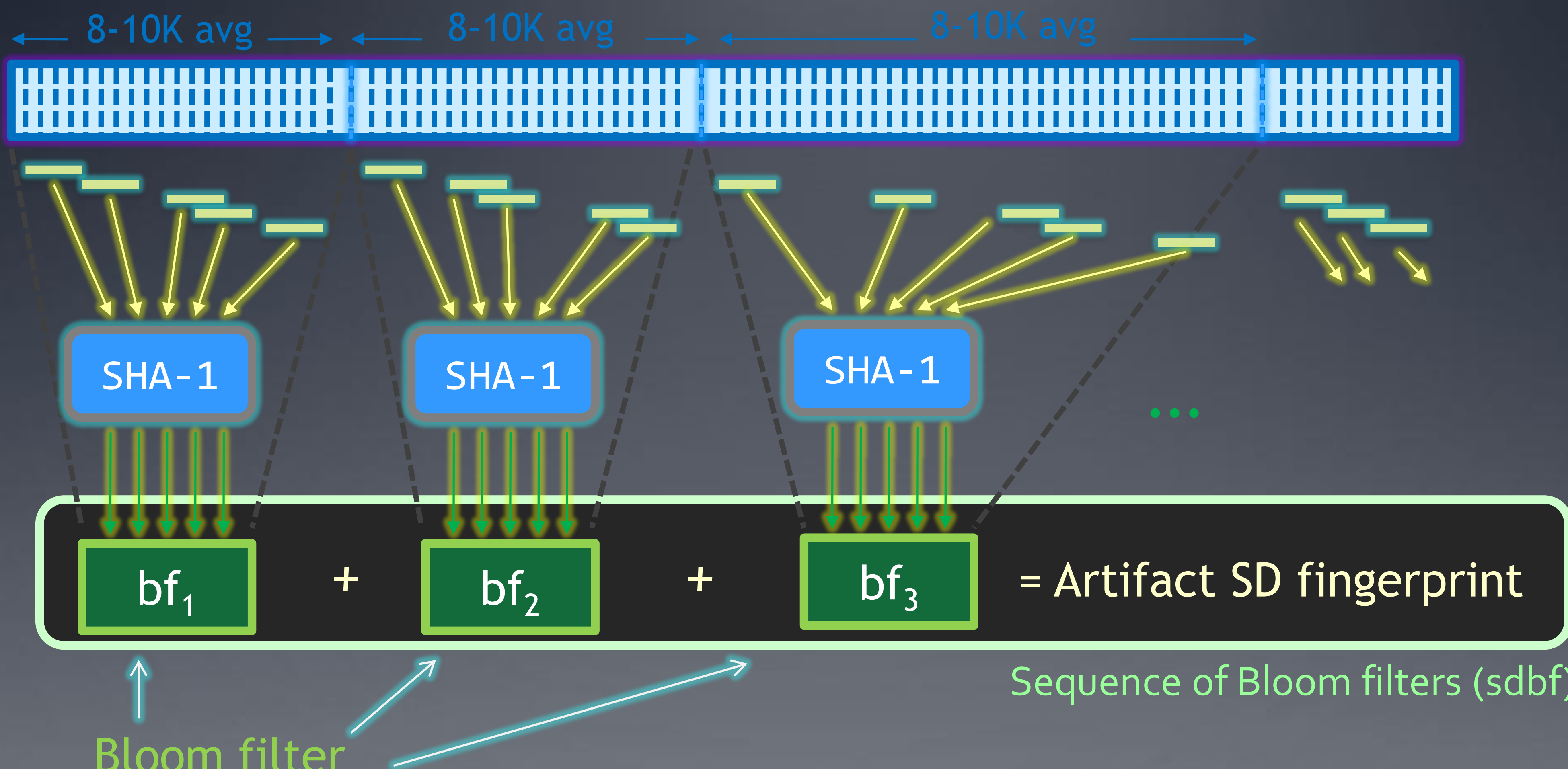


Generating sdhash fingerprints (3)

Feature Selection Process

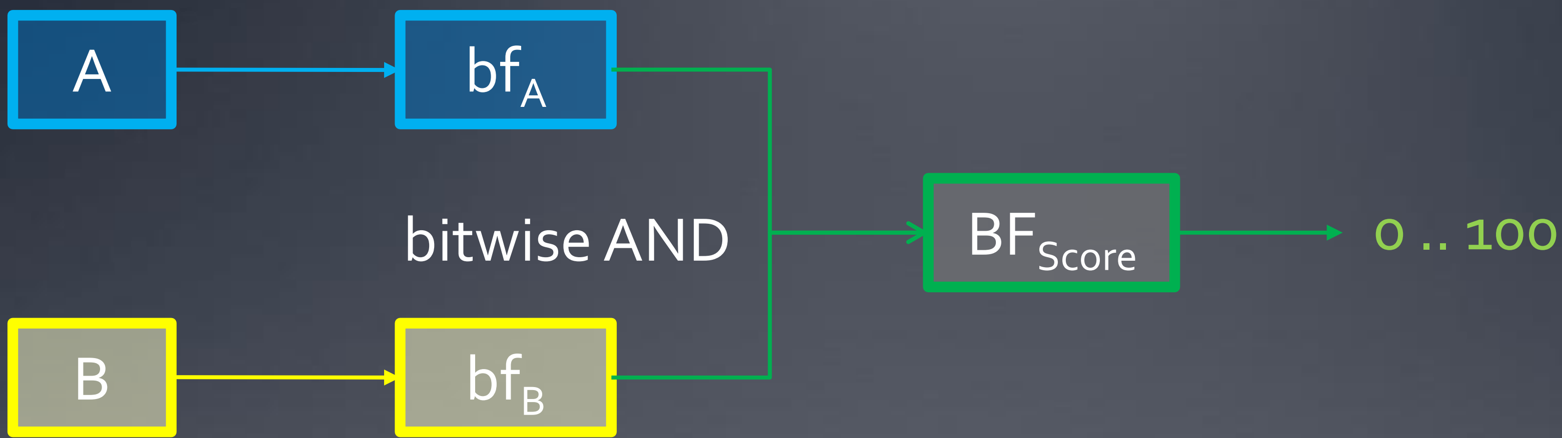


Generating sdbf fingerprints (4)



- local SD fingerprint
- 256 bytes
- up to 128/160 features

Bloom filter (BF) comparison

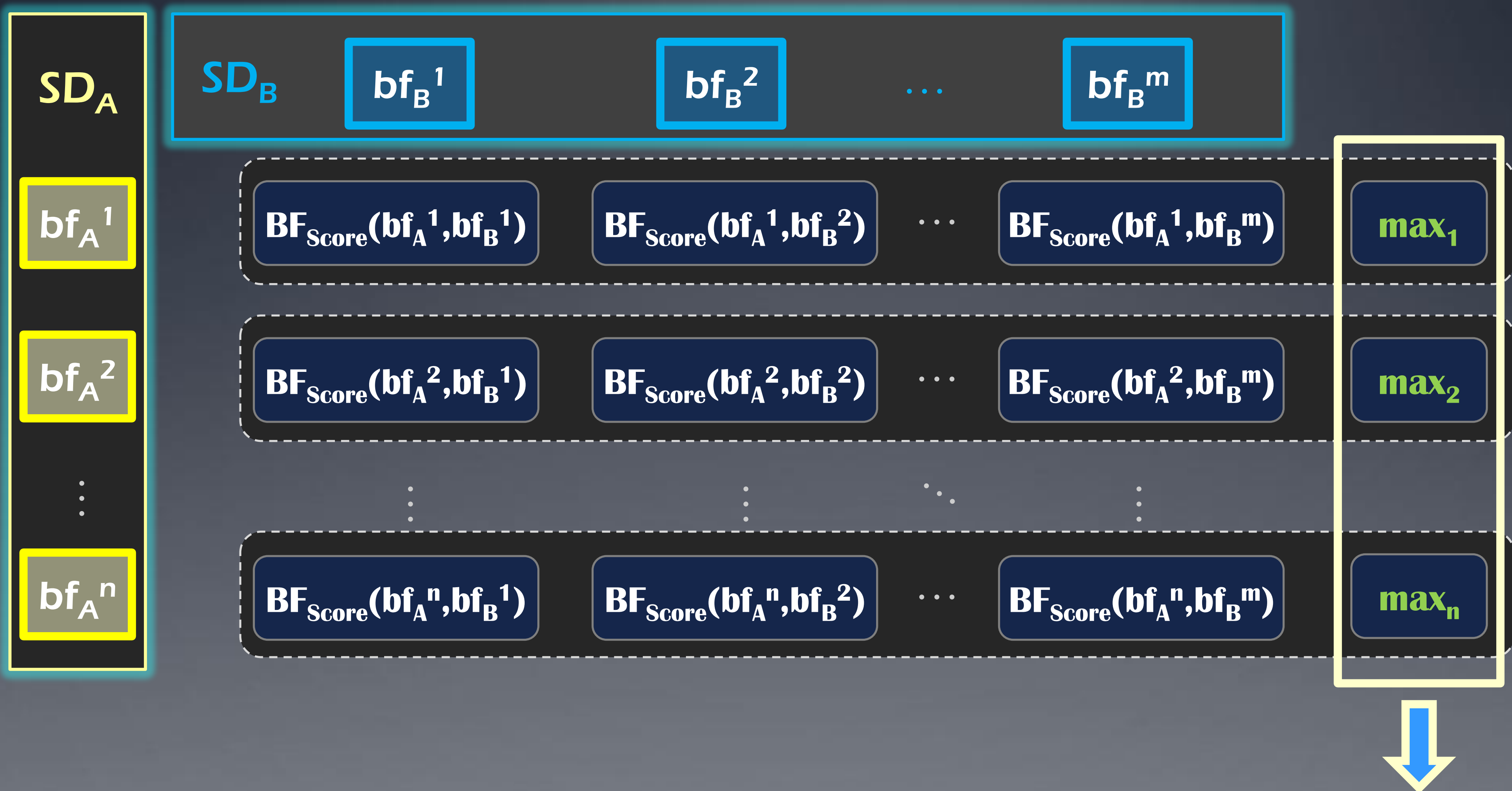


Based on BF theory,
overlap due to chance is analytically predictable.

Additional BF overlap is proportional to overlap in features.

BF_{Score} is tuned such that $BF_{Score}(A_{random}, B_{random}) = 0$.

SDBF fingerprint comparison

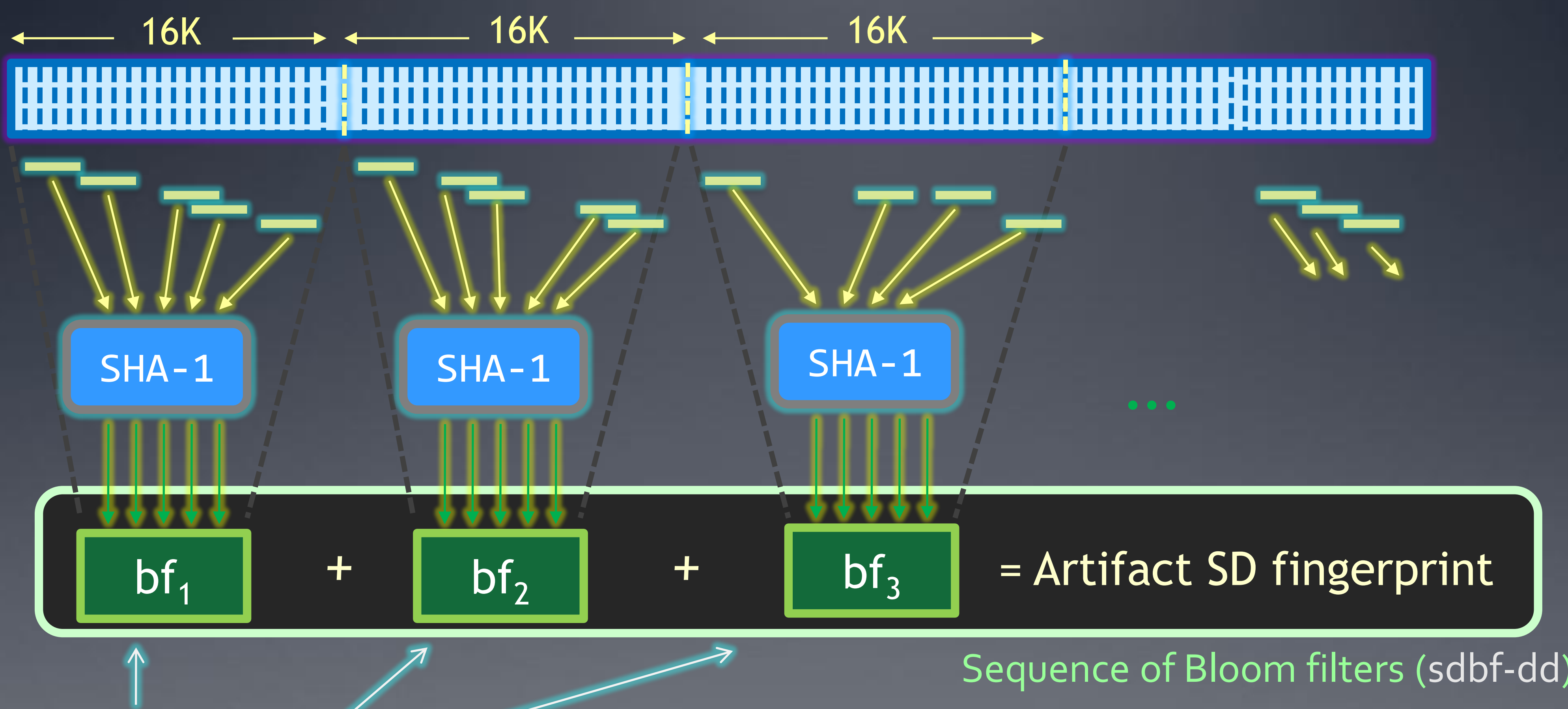


$$SD_{Score}(A,B) = \text{Average}(max_1, max_2, \ldots, max_n)$$

Scaling up:

Block-aligned digests & parallelization

Block-aligned similarity digests (sdbf-dd)



Bloom filter
- local SD fingerprint
- 256 bytes
- up to 192 features

Advantages & challenges for block-aligned similarity digests (sdbf-dd)

- Advantages
  - Parallelizable computation
  - Direct mapping to source data
  - Shorter (1.6% vs 2.6% of source)
  - ➔ Faster comparisons (fewer BFs)
- Challenges
  - Less reliable for smaller files
  - Sparse data
  - Compatibility with sdbf digests
- Solution
  - Increase features for sdbf filters: 128➔ 160
  - Use 192 features per BF for sdbf-dd filters
  - Use compatible BF parameters to allow sdbf ⇔ sdbf-dd comparisons

sdhash 1.7: sdbf vs. sdbf-dd accuracy

Query size	FP rate	TP rate	Query size	FP rate	TP rate
1,000	0.1906	1.000	2,000	0.0006	0.997
1,100	0.0964	1.000	2,200	0.0005	1.000
1,200	0.0465	1.000	2,400	0.0001	1.000
1,300	0.0190	1.000	2,600	0.0001	0.997
1,400	0.0098	1.000	2,800	0.0000	1.000
1,500	0.0058	1.000	3,000	0.0000	0.999
1,600	0.0029	0.999	3,200	0.0000	0.998
1,700	0.0023	0.999	3,400	0.0000	0.998
1,800	0.0013	0.999	3,600	0.0000	1.000
1,900	0.0010	0.998	3,800	0.0000	0.998