Wirespeed: Extending The Aff4 Container Format For Scalable Acquisition And Live Analysis

By

Bradley Schatz

Presented At

The Digital Forensic Research Conference

DFRWS 2015 USA Philadelphia, PA (Aug 9th - 13th)

Wirespeed: Extending the AFF4 forensic container format for scalable acquisition and live analysis

Dr. Bradley Schatz

Director, Schatz Forensic

DFRWS Conference 2015

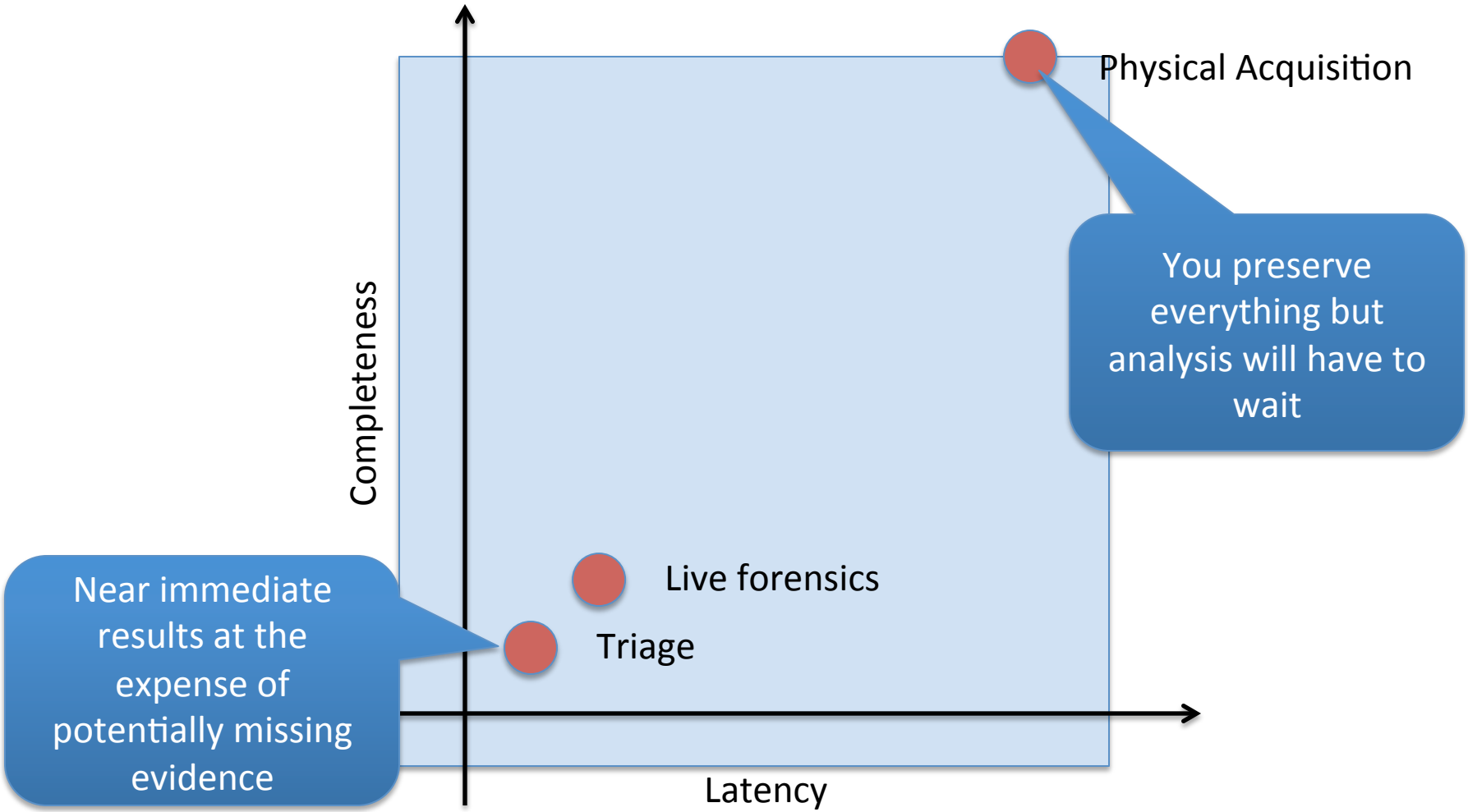
Overview

- The current approach to forensic acquisition is a bottleneck in the forensic process
- Propose additions to the AFF4 container format to support:
  - Partial acquisition
  - Acquisition at maximal I/O rates
- Empirical results of the proposed approach

Background

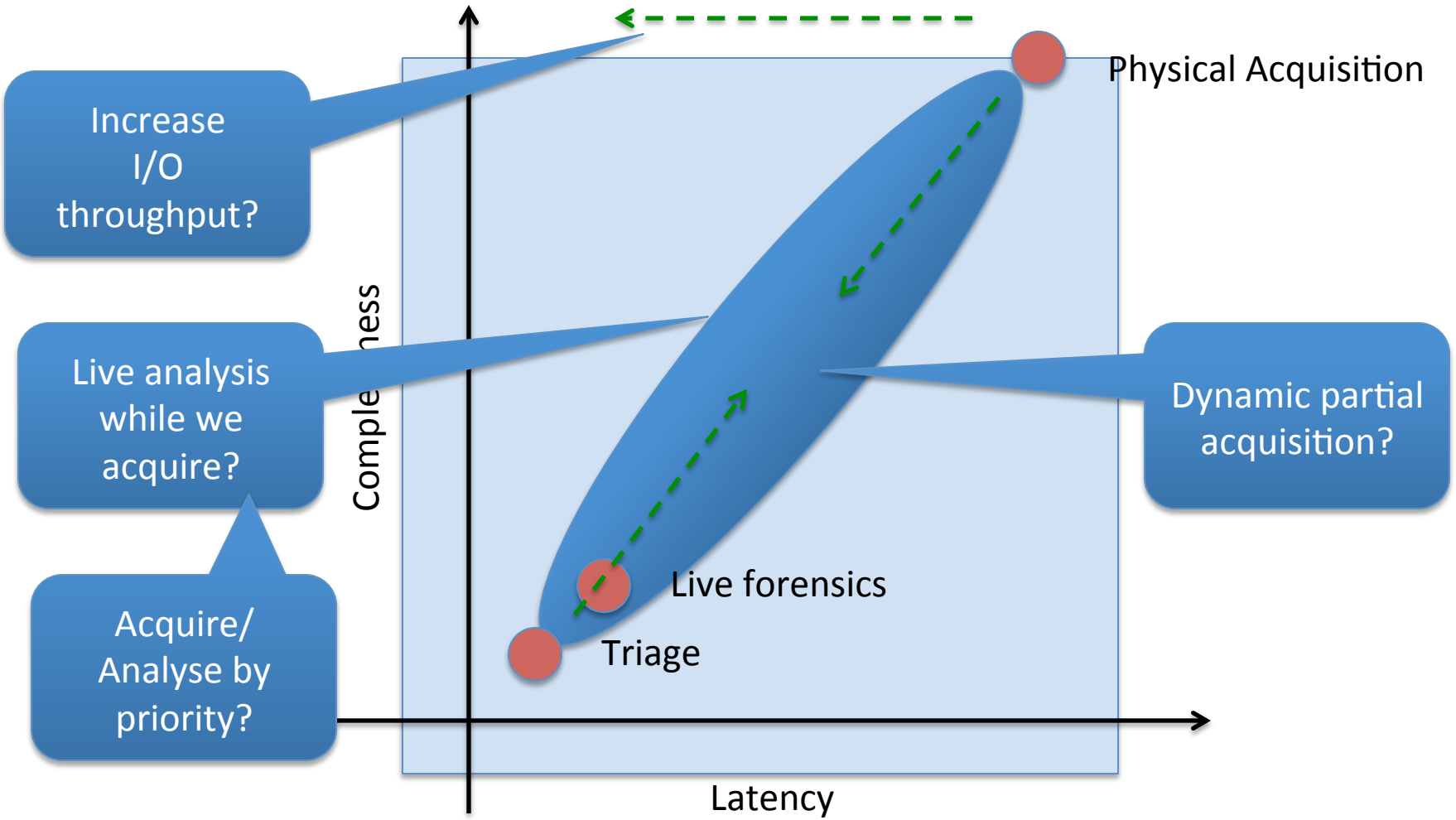
Pick one of the below

You can't have both



How can we reduce latency?

While maximising completeness



Why can't I have both?

The de-facto standard evidence containers are a limiting factor.

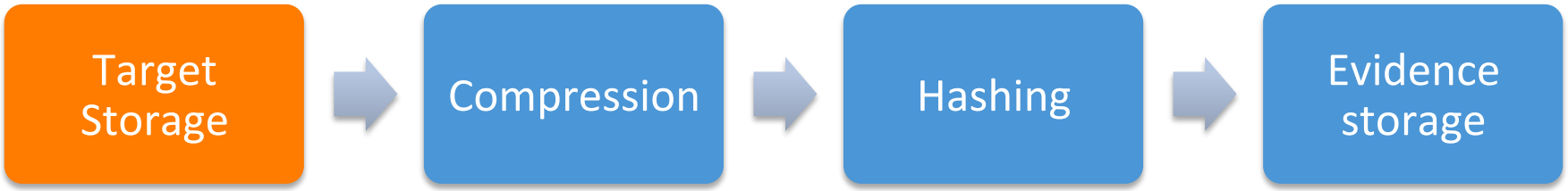
- Linear complete bit stream
  - All or nothing preservation choice
  - Prevents non-linear/prioritised preservation of evidence
- Heavyweight compression (Inflate)
  - Limiting factor on current CPU's (even with multi-core threading)
- Linear bytestream hash
  - Prevents non-linear/prioritised preservation
  - Hashing is limiting factor at high bitrates and with low CPU resources
- Single storage device
  - Evidence output device I/O rate often limiting factor
- Logical imaging
  - Missing raw filesystem and volume metadata

I/O Throughput

Is there actually a problem with throughput here?

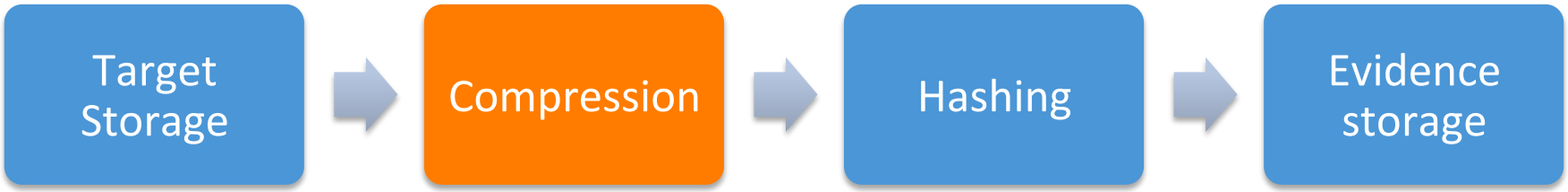
- Research publications
  - FastDD <= 110 MB/s [Bertasi & Zago 2013]
- Practitioner reports
  - Low 100's MB/s [Zimmerman 2013]
- Vendor marketing
  - Hardware devices promising 250MB/s [Tableau 2014]

I/O throughput in acquisition is a systems problem



Target Storage	Max Read
Current generation 3.5” 7200rpm SATA	200 MB/s
Intel 730 SSD	550 MB/s
Macbook Pro 1TB (real data)	100MB/s
Macbook Pro 1TB (sparse)	1 GB/s
RAID 15000rpm SAS	> 1 GB/s

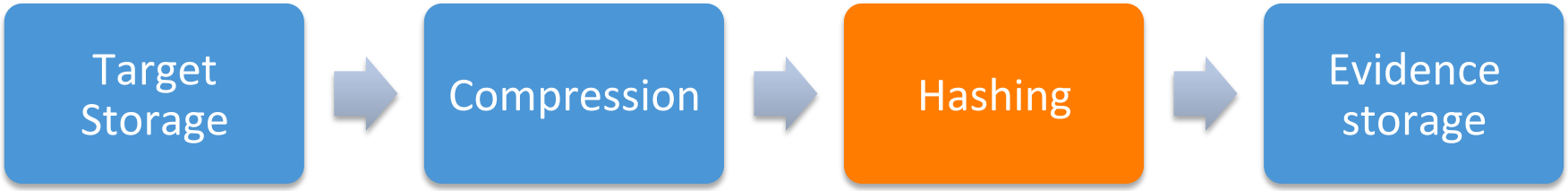
Inflate compression is costly in CPU resources



Algorithm	Throughput MB/s*
Inflate	39.42
Snappy (Google BigTable/MapReduce)	1,405.42
LZO (ZFS)	1,538.31

*Single core of quad core i7-4770 3.4Ghz

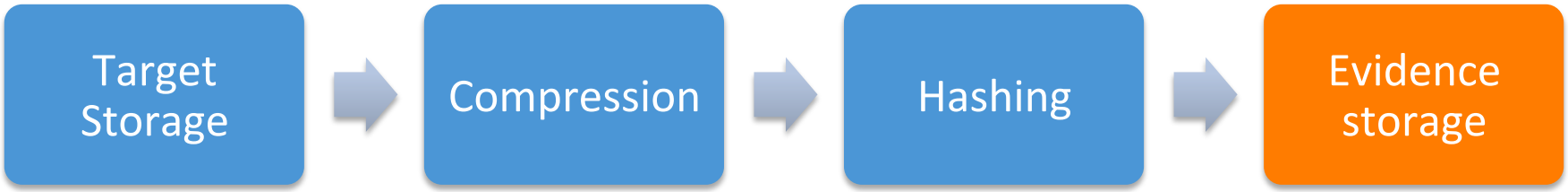
Hashing is the next most expensive acquisition operation



Algorithm	Throughput MB/s
SHA1	619.23
MD5	745.65
Blake2b	601.87

*Single core of quad core i7-4770 3.4Ghz

I/O Rate of acquisition is a systems problem



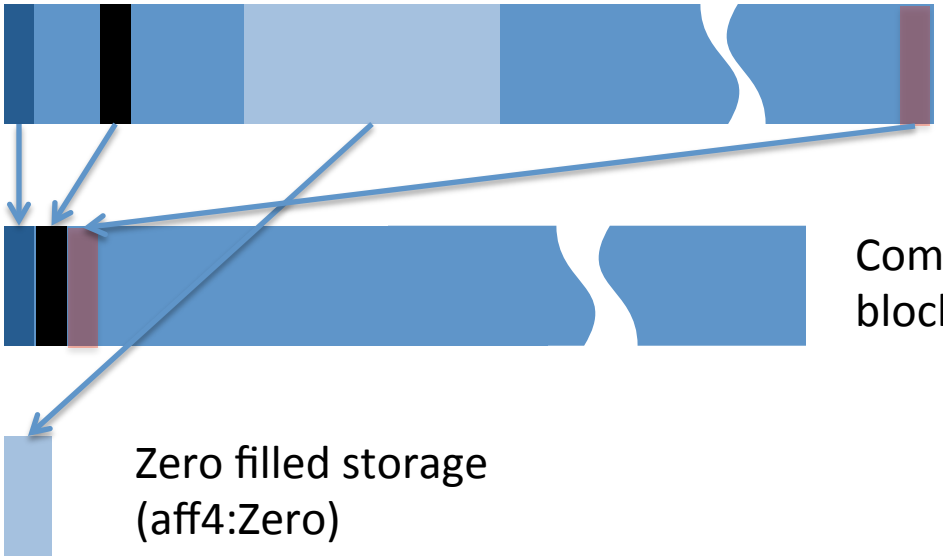
Output	Gb/s	MB/s
SATA3	6	600
USB3	5	500
Commodity SATA 7200 rpm		*200*
Gigabit Ethernet	1	100
USB2	.48	48

Our proposal

Extensions to AFF4

AFF4 in a nutshell

- Virtual block storage (Maps)
  - Non-linear, composable
- Compressed block storage (Streams)
- Globally unique naming scheme
- There is an object representing each entity



/mapID/map
/mapID/index

/streamID/00000
/streamID/00000/index
/streamID/00001
/streamID/00001/index
…

Faster compression
Symbolic sections

- Extension: we define virtual bytestreams "*aff4:SymbolicStream00*" to "*aff4:SymbolicStreamFF*"
- Synonyms aff4:Zero and aff4:FF
  - Use case: Zero filled sectors and erased flash blocks

AFF4 Map example

```
0,0,aff4://0466b8fb-9af0-4ef2-b36c-8b0d90fc0ac2>
4096,0,aff4:SymbolicStreamFF
8192,4096, aff4://0466b8fb-9af0-4ef2-b36c-8b0d90fc0ac2>
```

Partial acquisition

- Challenge: Representing what we didn't acquire
- Extensions: we define two symbols
  - aff4:UnreadableData : *Blocks that we tried to read but couldn't*
  - aff4:UnknownData : *Blocks that we never even tried to read*

AFF4 Map example

```
0,0,aff4://0466b8fb-9af0-4ef2-b36c-8b0d90fc0ac2>
4096,0,aff4:UnknownData
8192,4096, aff4://0466b8fb-9af0-4ef2-b36c-8b0d90fc0ac2>
```

Faster compression

More speed-efficient algorithms

- Extension: the storage stream now has property called "*aff4:compressionMethod*"

AFF4 Stream example

```
<aff4://0466b8fb-9af0-4ef2-b36c-8b0d90fc0ac2> a
aff4:stream ;
       aff4:CompressionMethod <http://code.google.com/p/
snappy/> ;
       aff4:chunk_size "32768"^^xsd:int ;
       aff4:size "294912"^^xsd:long ;
```

Faster hashing

Non-linear parallelised block hashing

- Our proposal:
  - Deprecate the linear bytestream hash
  - Parallelise hashing by using segment hashes
  - Hashing symbolic chunks is a waste of CPU resources – hash the map instead
  - Take a singular hash of the above hashes

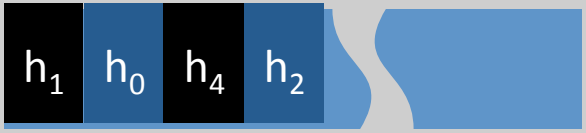
Faster hashing

Non-linear parallelised block hashing

Image (Map)

mapPointHash=sha256(mapID/map)
mapIndexHash=sha256(mapID/index)

Stream B



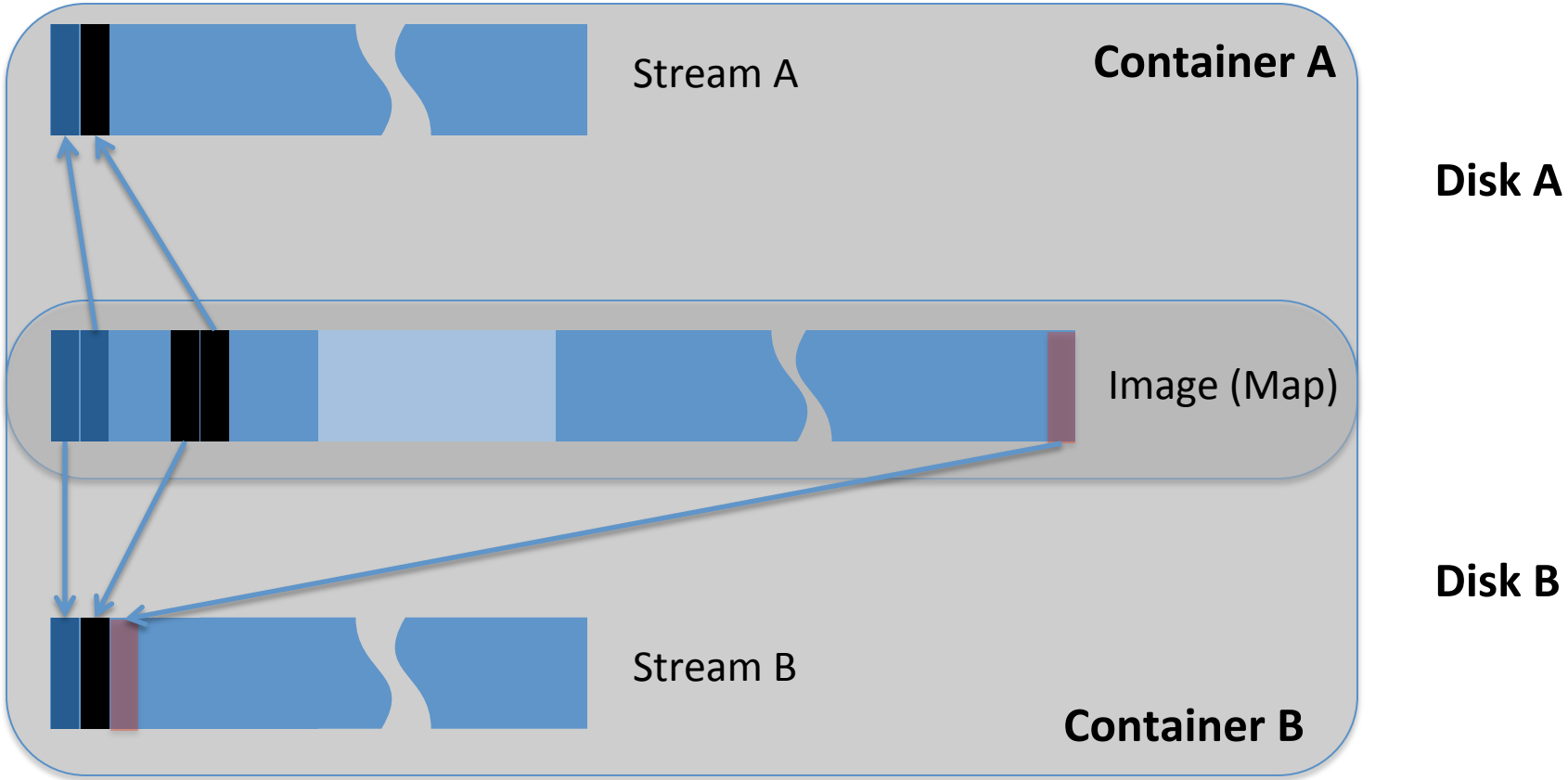
Stream B
block hashes

blockHashesHash=sha256(h_1,h_0,h_4,h_2…)

blockHash=sha256(blockHashesHash$_0$ .. blockHashesHash$_n$. mapPointHash . mapIndexHash)

Aggregate Output Channels

- Use the aggregate I/O capacity of the device

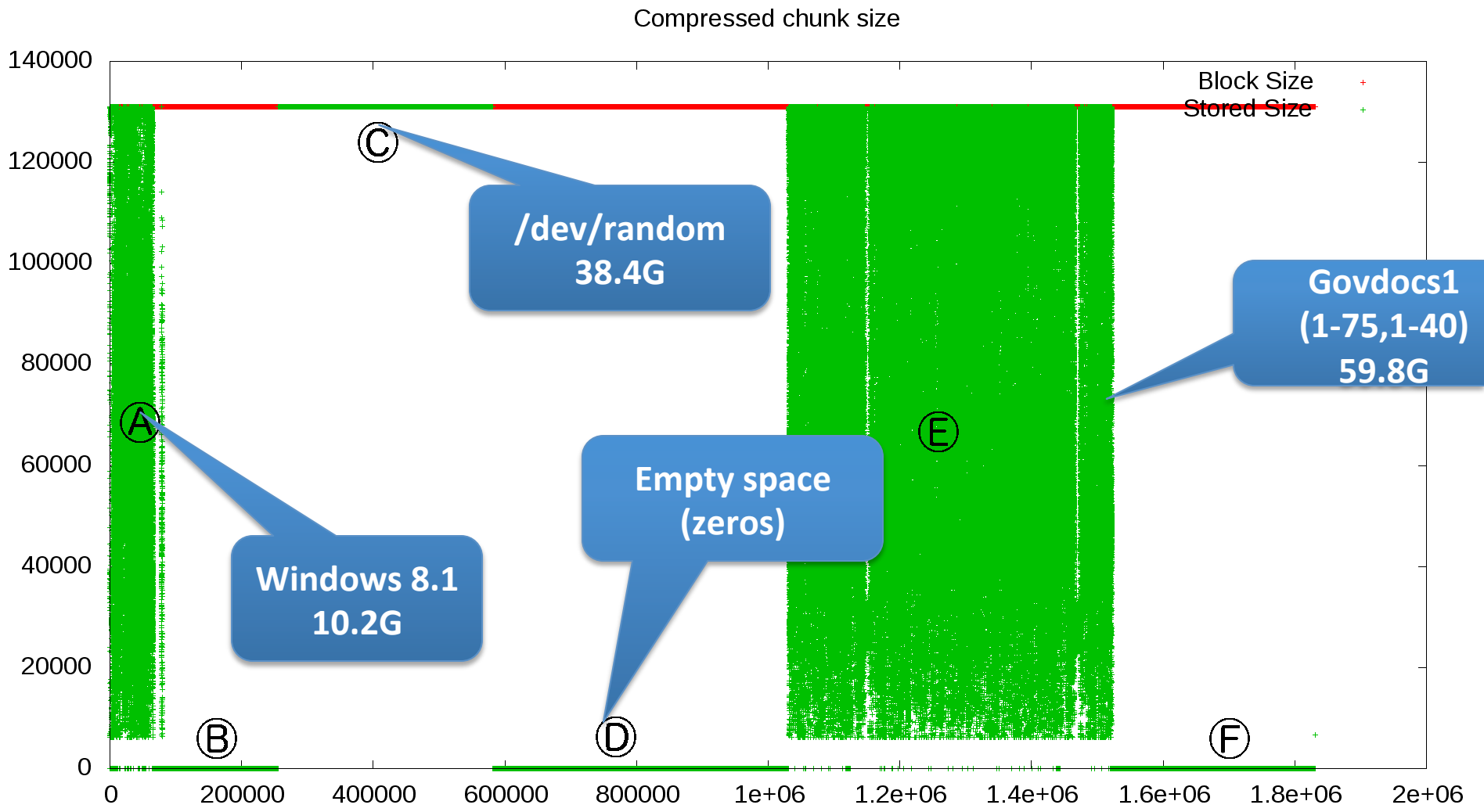


Experimental validation

Methodology

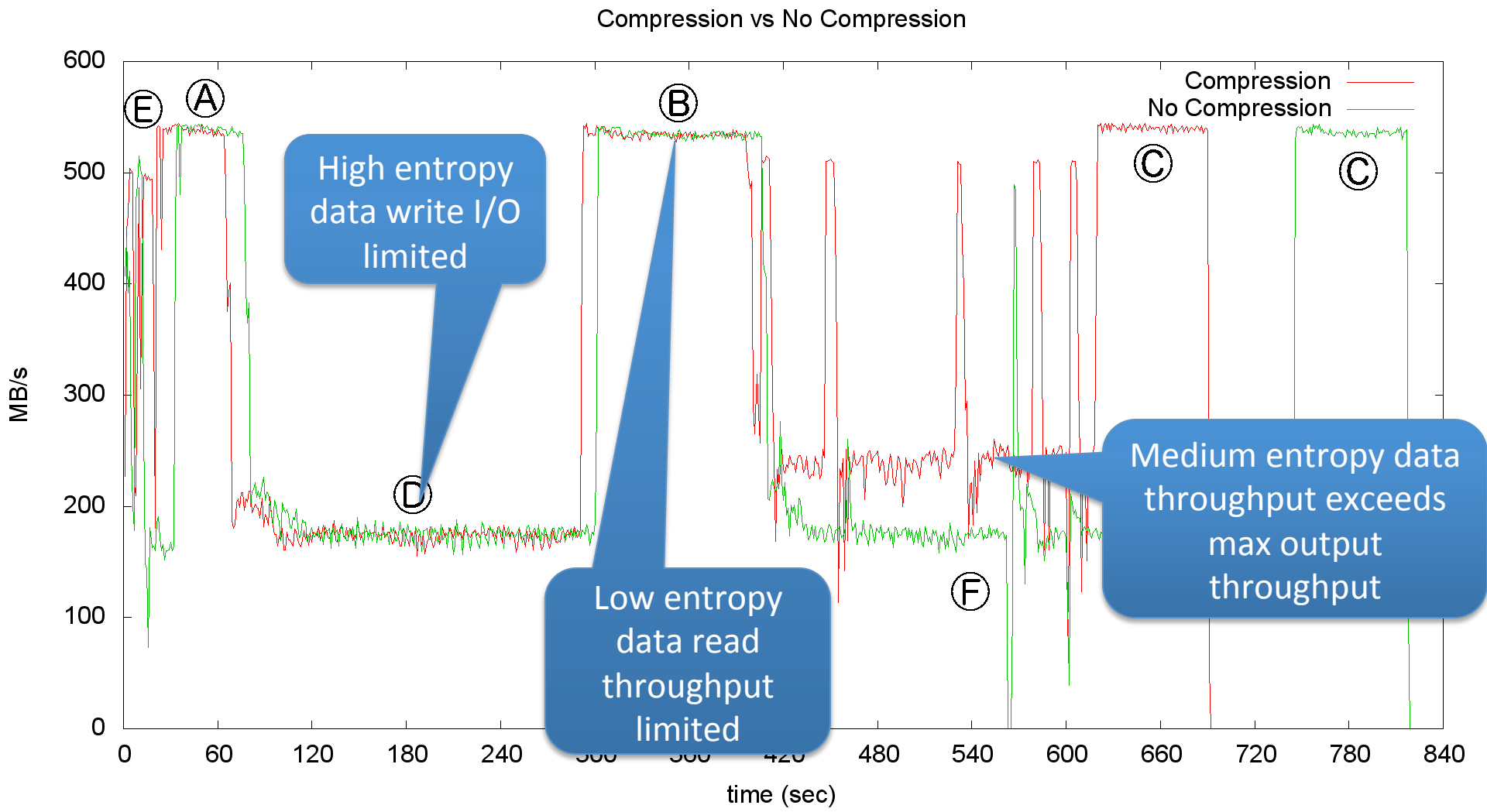
- We built a forensic acquisition/analysis system
  - Non-linear, partial acquisition & live analysis (called Wirespeed)
- Prepare testbed
  - Target disk: Intel 730 240G SSD (max read 530 MB/s)
  - Destination disks: Toshiba 2TB 7200RPM SATA (max write near 200 MB/s)
  - Computer 1: 4 core i7-4770R 3.20GHz
  - Computer 2: 2 core i5-3337U 1.80GHz
- Prepare test sample
- Test, varying on
  - CPU
  - IO Channels

Test standard composition
Stored block size –v- LBA address



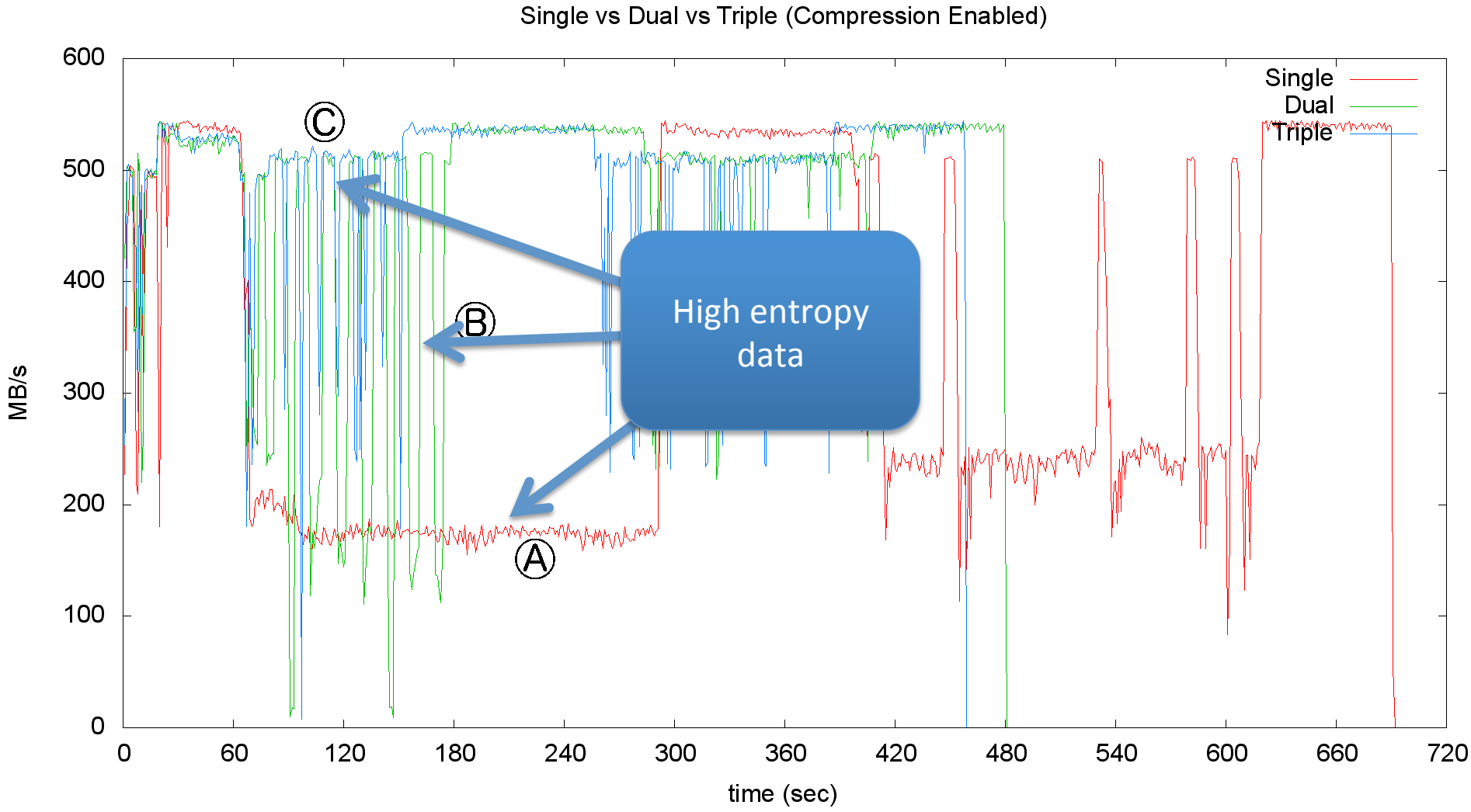
Compression is faster than raw

Single output drive



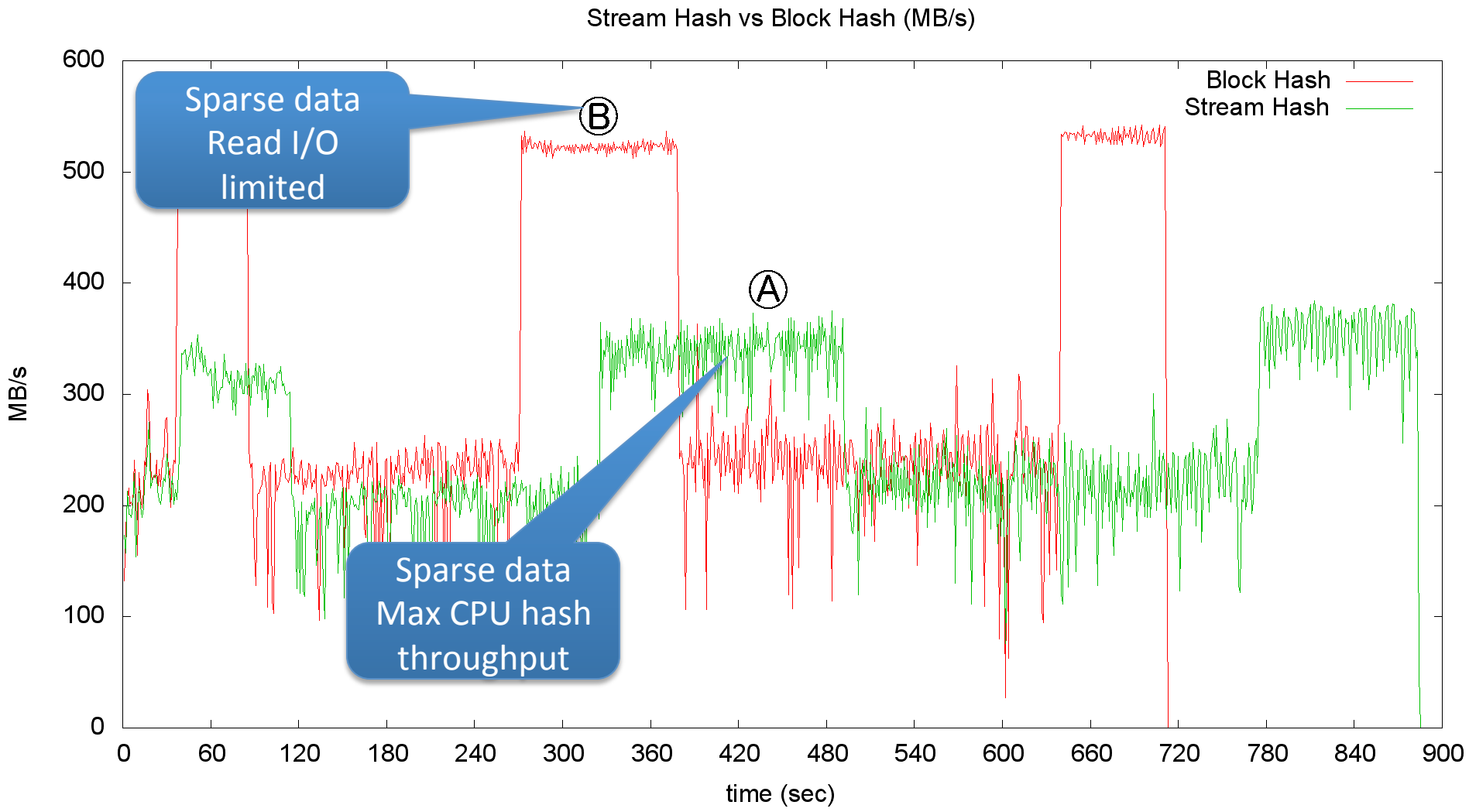
Multiple output channels increases throughput

Especially for uncompressible data

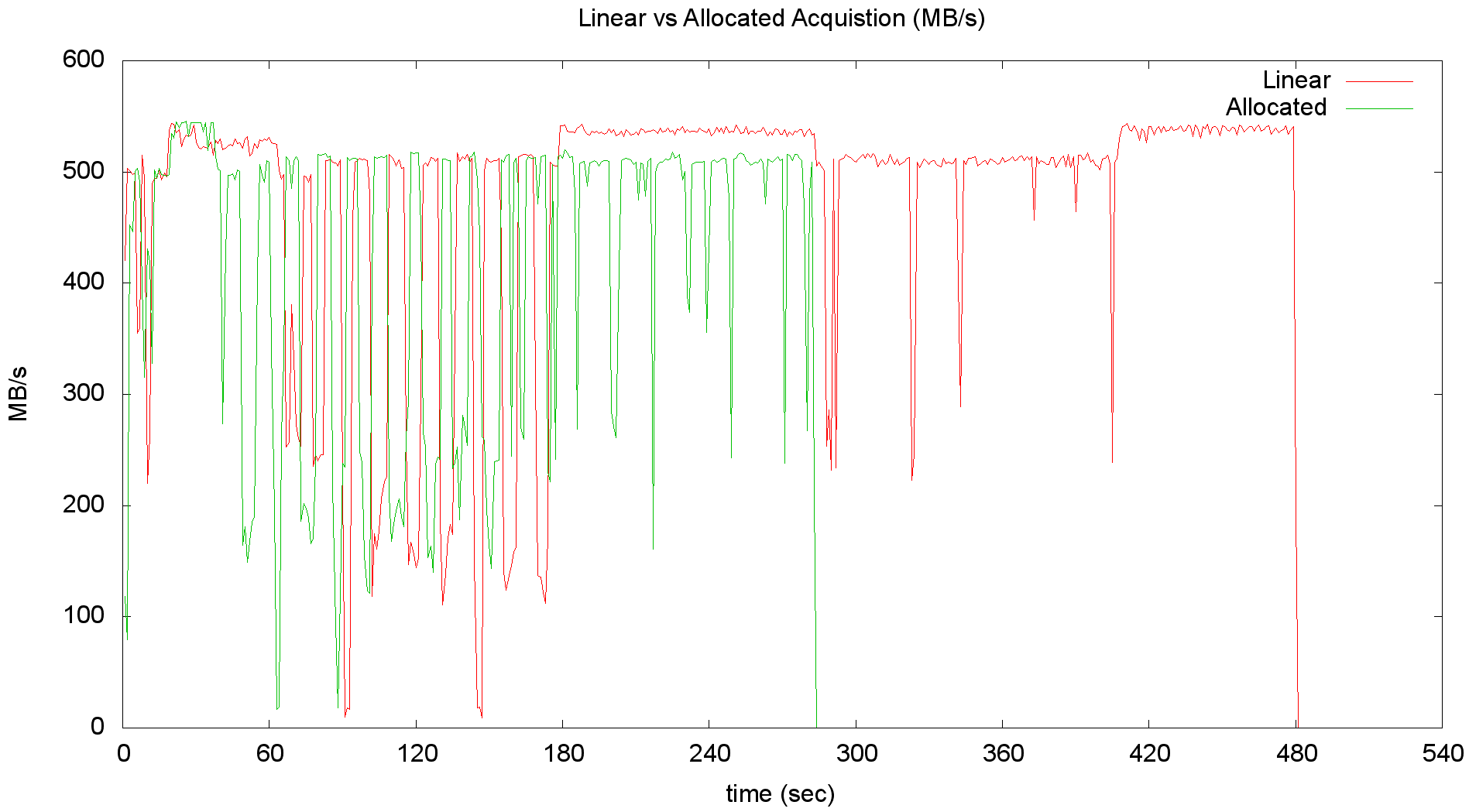


Block based hashing beats linear stream hashing with low powered multicore CPU's

Dual core i5



Non-linear partial imaging gives significant gains over linear



The proposed approach gives significant throughput gains over current implementations.

Acquisition application	I7-4770R 3.2 GHz system	I5-3337U 1.8GHz system
FTK Imager	20:10 (198 MB/s)	37:38 (106MB/s)
X-Ways Forensics	13:58 (286 MB/s)	33:23 (120 MB/s)
Wirespeed (linear)	11:29 (384 MB/s)	15:08 (264 MB/s)

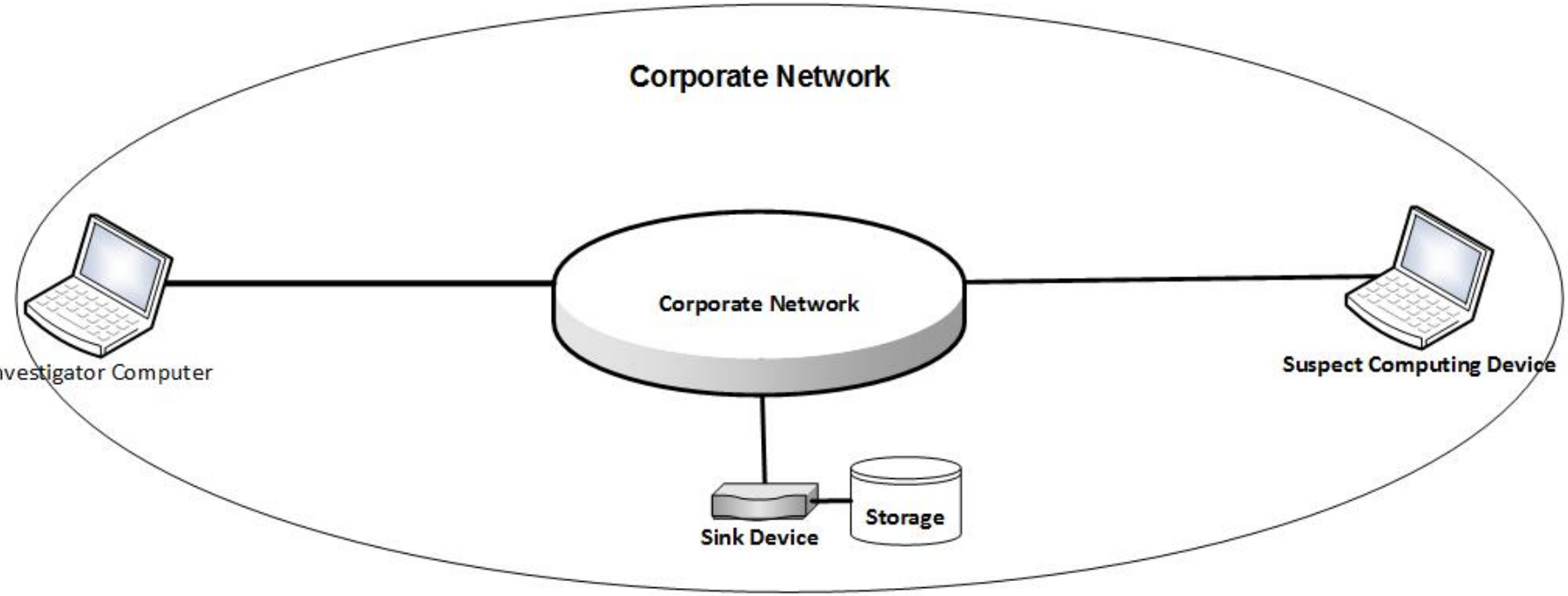
Comparative acquisition speeds

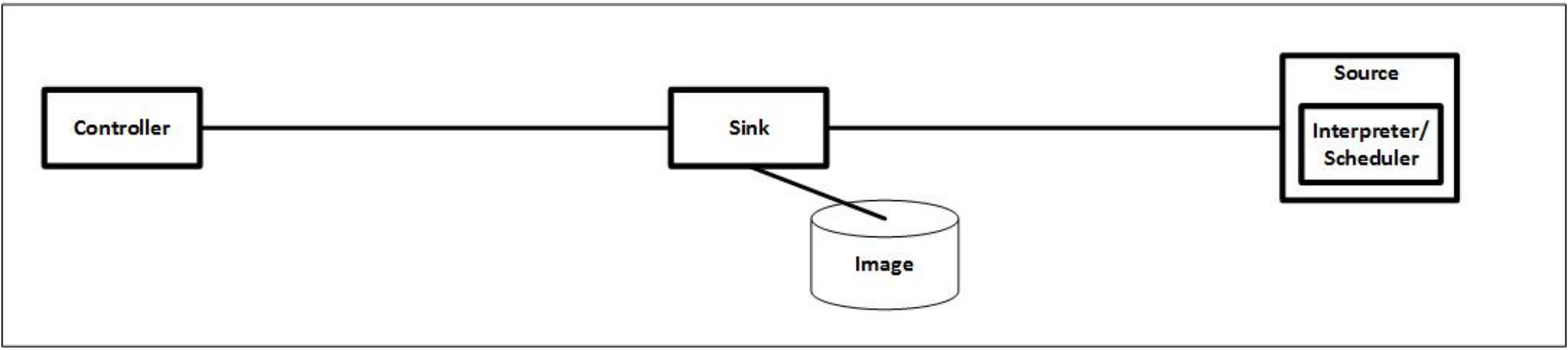
	1 Stripe	2 Stripes	3 Stripes
Wirespeed (linear)	11:29 (384 MB/s)	8:00 (500 MB/s)	7:30 (533 MB/s)
FTK Imager	20:10 (198 MB/s)	N/A	N/A
X-Ways Forensics	13:58 (286 MB/s)	N/A	N/A
Wirespeed (allocated)	8:21 (229 MB/s)	4:42 (408 MB/s)	4:17 (447 MB/s)

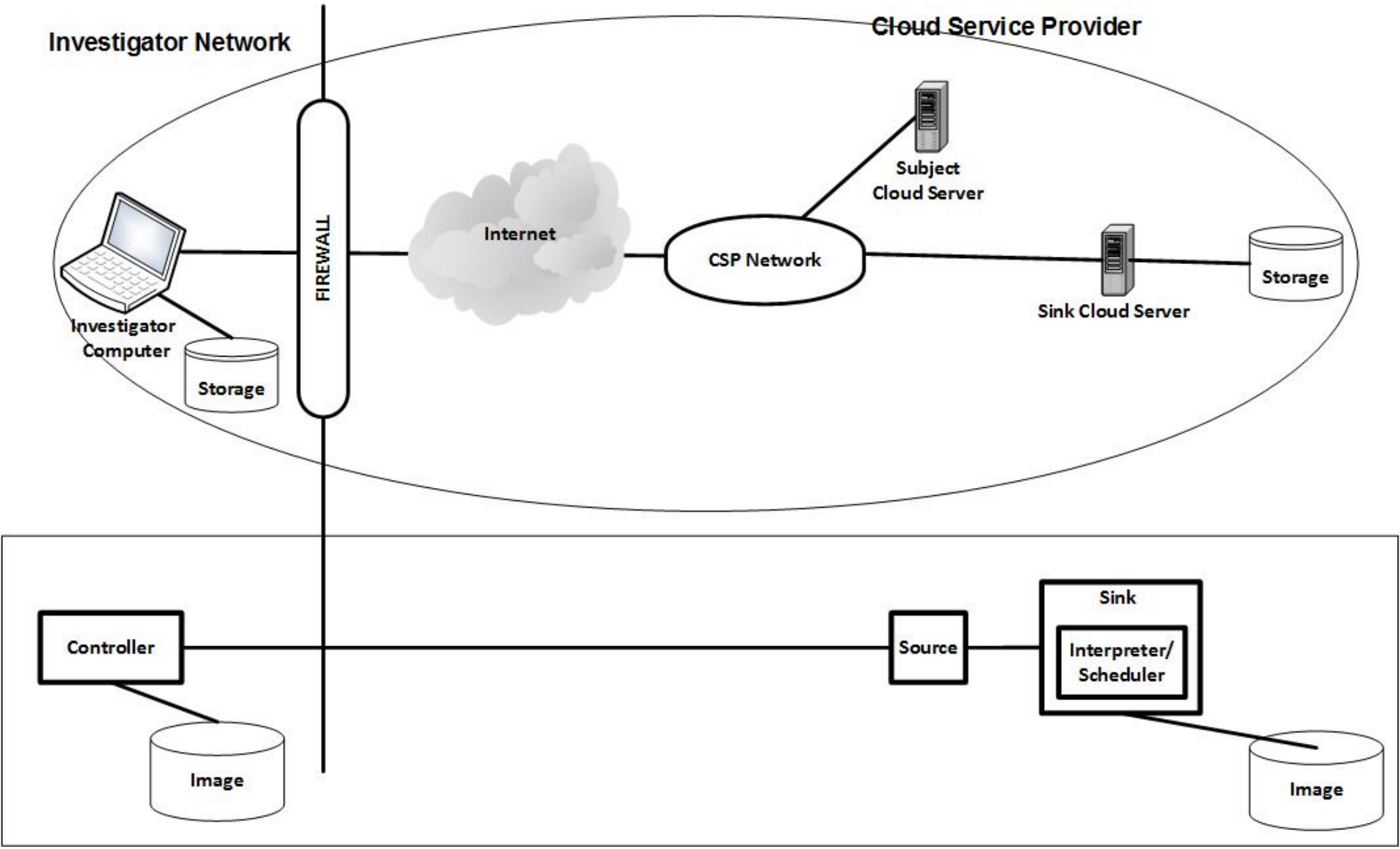
Conclusions

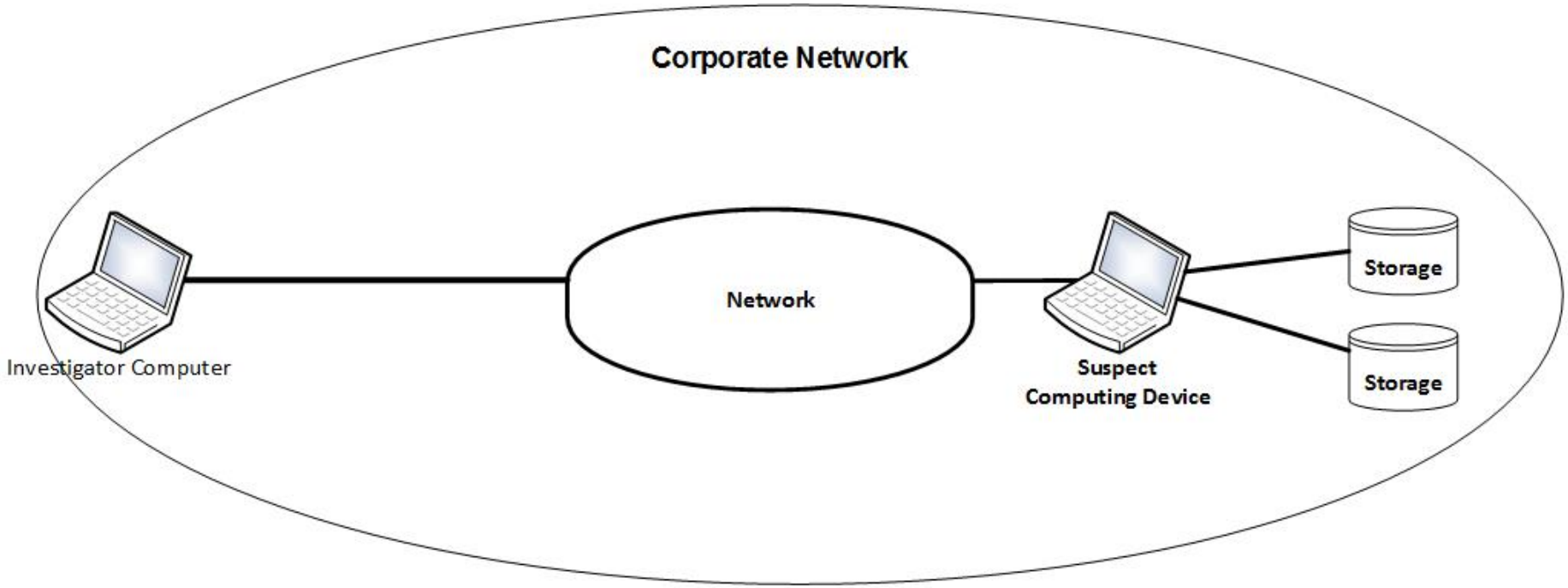
Conclusion

- Existing image formats are a limitation
  - Linear byte stream hash
  - Inflate algorithm
- Extensions to the AFF4 format proposed
  - Faster hashing and compression
  - Partial images
  - Exploitation of aggregate IO channels
- Proof of concept demonstrated significant throughput gains and improved latency
- Our implementation is available at:
  - http://wirespeed.io






Investigator Network
Cloud Service Provider
Subject
Cloud Server
FIREWALL
Internet
CSP Network
Storage
Sink Cloud Server
Investigator
Computer
Storage
Controller
Source
Sink
Interpreter/
Scheduler
Image
Image


Corporate Network
Network
Investigator Computer
Suspect
Computing Device
Storage
Storage


Source
Interpreter/
Scheduler
Controller
Sink
Image
Image