

A large yellow rubber duck is floating in the water in the foreground. In the background, a dense city skyline with various skyscrapers is visible under a cloudy sky. One prominent building has a red sign that says "SHUN TAK".

SURF

How to analyse a ddos quackly

(with duck all effect on memory)

Remco Poortinga – van Wijnen
remco.poortinga@surf.nl

National anti-DDoS Coalition

- Organisations from different sectors collaborating in the fight against DDoS
 - Different working groups: technical (clearing house), drills, legal, communication, ...
- Doing DDoS drills together
- Exchanging information
- <https://www.nomoreddos.org/>



Belastingdienst



Ministerie van Volksgezondheid,
Welzijn en Sport



Agentschap Telecom
Ministerie van Economische Zaken
en Klimaat



Stichting Digitale
Infrastructuur Nederland



Logius
Ministerie van Binnenlandse Zaken en
Koninkrijksrelaties



UNIVERSITY
OF TWENTE.



Nationaal Cyber Security Centrum
Ministerie van Veiligheid en Justitie



Exchanging (technical) DDoS information

- DDoS attacks typically combine techniques
 - 'Attack vectors'
 - DNS reflection, chargen, UDP flood, NTP, ...
- 'Fingerprints' describe/summarise (these aspects of) an attack →
- Analysis by ddos_dissector tool

```
{
  "attack_vectors": [
    {
      "service": null,
      "protocol": "UDP",
      "fraction_of_attack": 1.0,
      "source_port": "random",
      "destination_ports": {
        "3650": 1.0
      },
      "tcp_flags": null,
      "nr_packets": 59770,
      "nr_megabytes": 2,
      "time_start": "2022-10-26T11:32:58.263795+00:00",
      "duration_seconds": 18,
      "source_ips": [
        "109.74.195.132",
        "97.107.135.252",
        "198.74.49.28",
        "172.105.209.31",
        "172.105.54.184"
      ],
      "ethernet_type": {
        "IPv4": 1.0
      },
      "frame_len": {
        "42": 1.0
      },
      "fragmentation_offset": {
        "0": 1.0
      },
      "ttl": {
        "55": 0.805,
        "53": 0.195
      }
    }
  ],
  "tags": [
    "UDP",
    "UDP flood attack"
  ],
  "key": "f2b689051c7a22dff37ff663f47b4133",
  "time_start": "2022-10-26T11:32:58.263795+00:00",
  "time_end": "2022-10-26T11:33:16.398928+00:00",
  "duration_seconds": 18,
  "total_packets": 59770,
  "total_megabytes": 2,
  "total_ips": 4,
  "avg_bps": 1115706,
  "avg_pps": 3320,
  "avg_Bpp": 42
}
```

| ddos_dissector original approach

- Split attack trace (pcap) into small chunks
 - Export chunks to CSV and load in dataframe (in parallel)
 - Concatenate dataframe chunks
 - Analyse attack
 - Infer attack target
 - filter data on target
 - Try to determine attack vectors
 - filter each attack vector
 - analyse/summarise attack vector
 - Create fingerprint
- operations with dataframe(s) in memory, e.g. determine most occurring protocol, source ports.
- Using dataframes means maximum size that can be analysed is limited by available memory.
 - Might using duckdb and parquet provide a viable alternative?

| Alternative duckdb_dissector approach

- Split attack trace (pcap) into chunks
- Export chunks to CSV (in parallel) (do not load in memory)
- Convert/concatenate CSVs chunks to parquet file
- Analyse attack (from parquet file on disk using duckdb)
 - Infer attack target
 - create view on target
 - Try to determine attack vectors
 - create view for each attack vector
 - analyse/summarise attack vector
- Create fingerprint

ddos_dissector vs duckdb_dissector

"Booters - An analysis of DDoS-as-a-Service Attacks" by José Jair Santanna, Roland van Rijswijk-Deij, Anna Sperotto, Rick Hofstede, Mark Wierbosch, Lisandro Zambenedetti Granville, and Aiko Pras. In Proceedings of 14th IFIP/IEEE Symposium on Integrated Network and Service Management (IM), May 11-15 2015, Ottawa, Canada.

https://www.simpleweb.org/wiki/index.php/Traces#Booters_-_An_analysis_of_DDoS-as-a-Service_Attacks

- Tests done using a standard linux (ubuntu) machine
 - AMD Ryzen 5 2600 (6 cores, 12 threads)
 - 16 GB of memory
 - 2 TB Seagate FireCuda 530
- Traces are simple screenshots of System Monitor tool
 - multiple stitched together for longer traces (>10 mins)

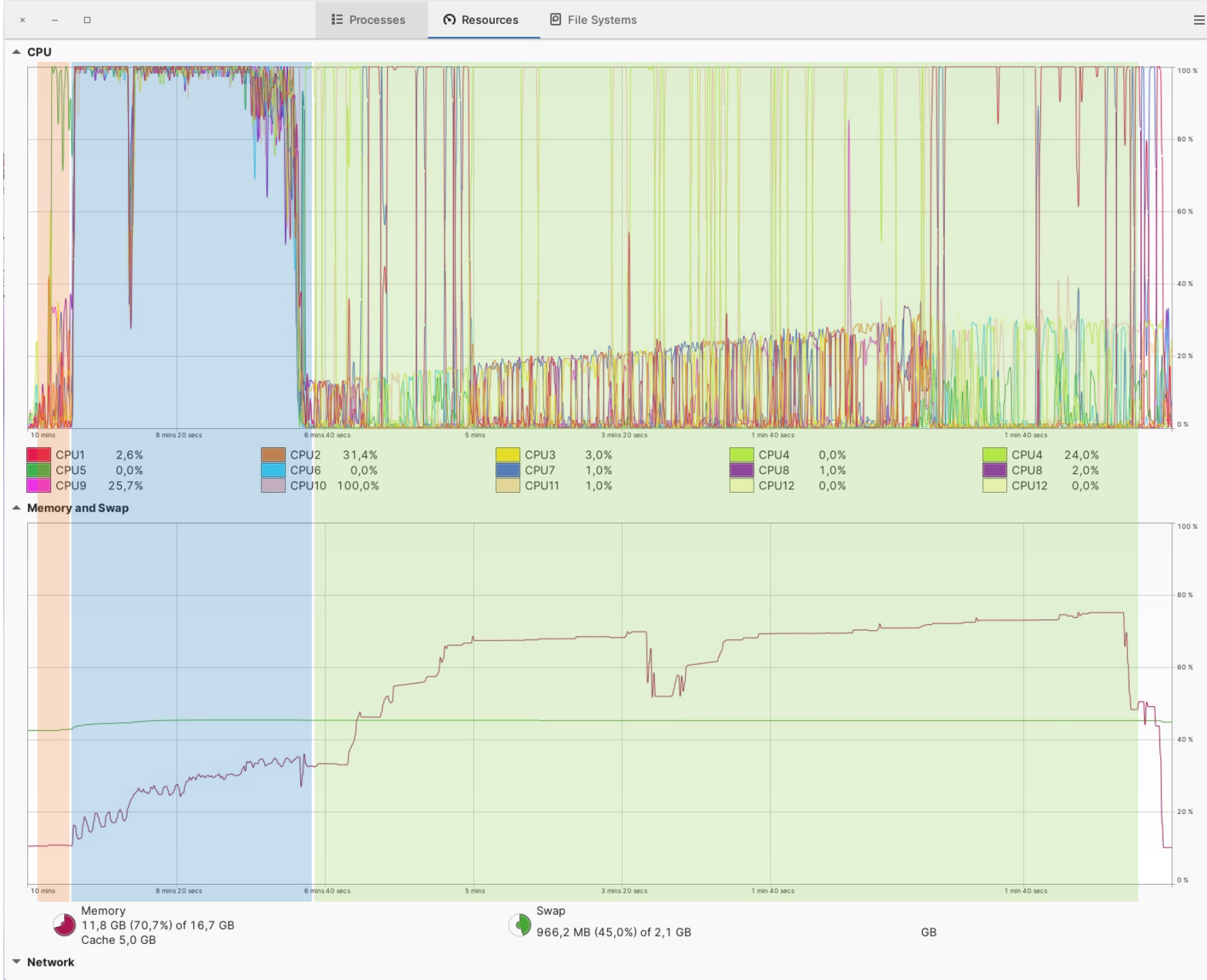
| anon-Booter8

- 7 GB pcap
- 5.75 million packets
- Chargen attack

ddos_dissector

12 minutes

- split pcap
- export CSV & load in memory
- analysis



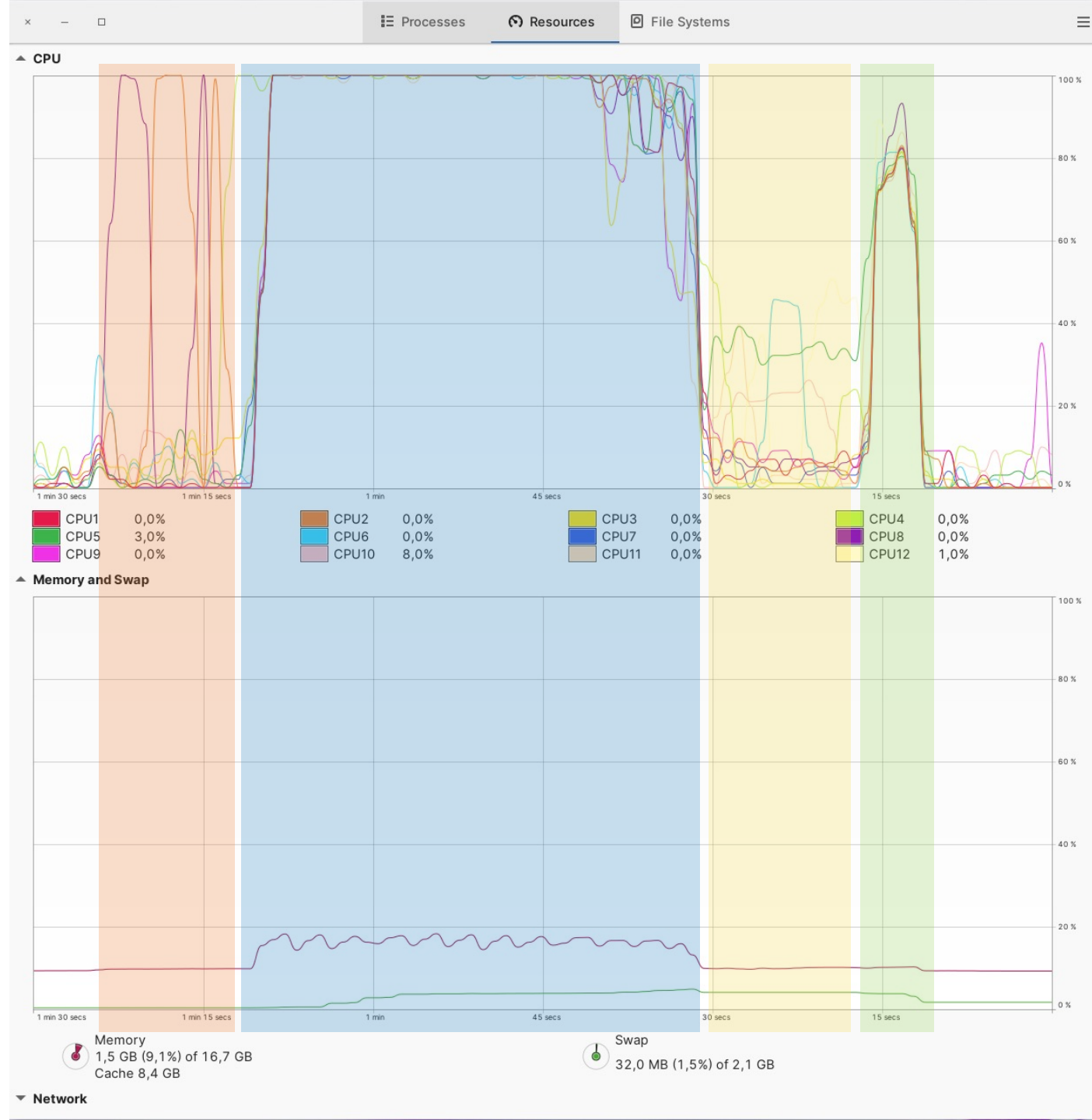
← CPUs

← MEM

duckdb_dissector

1 minute 14 seconds

- split pcap
- export CSV
- write parquet
- analysis

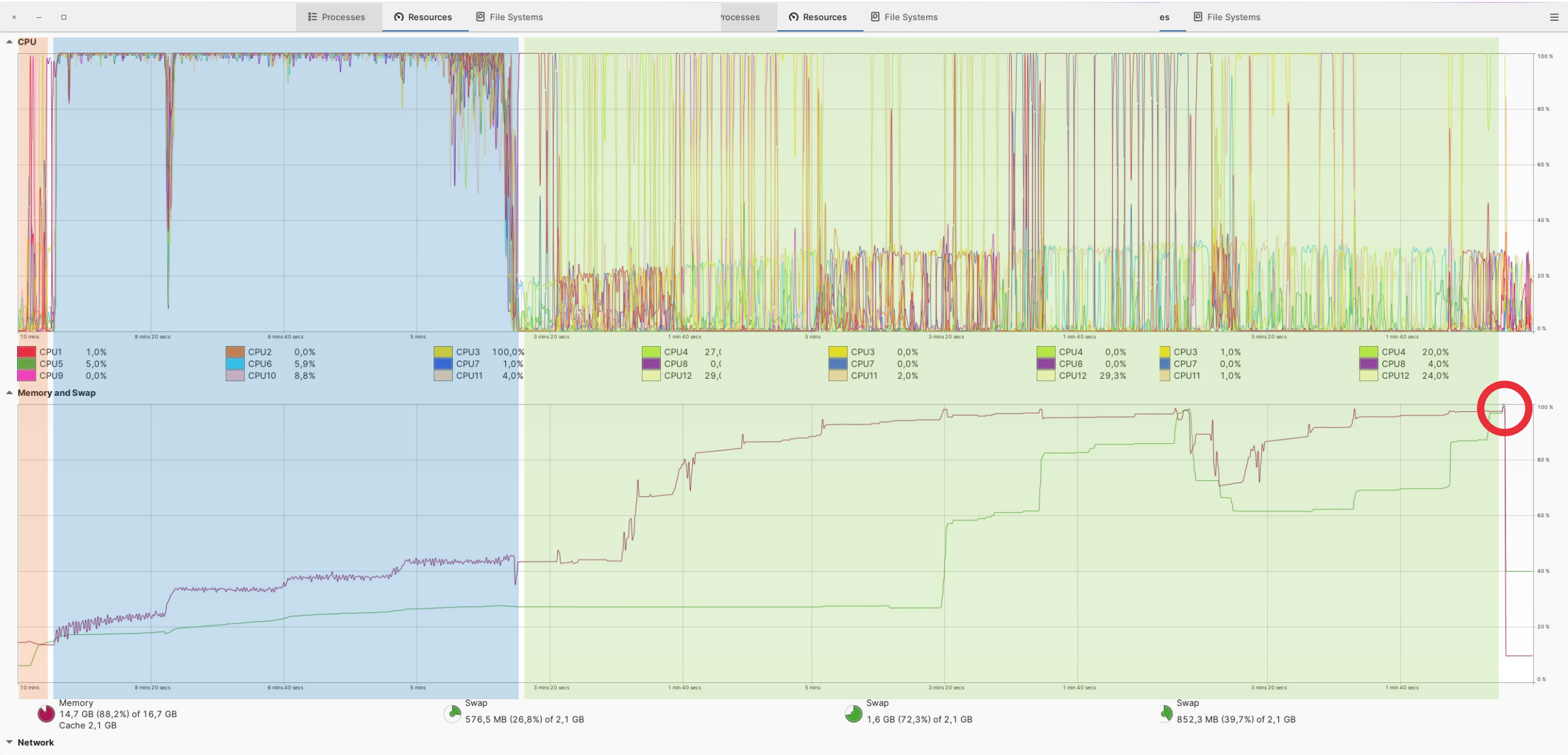


| anon-Booter2

- 8 GB pcap
- 9 million packets (appr. +50%)
- DNS attack

ddos_dissector

18 minutes DNF



duckdb_dissector
4 minutes 40 seconds



| anon-Booter4

- 50 GB pcap
- 36.6 million packets (appr. x4)
- DNS attack

30 minutes



| Conclusions

- Using duckdb/parquet for analysis is indeed a viable alternative
- Memory usage very consistent (and no longer an issue)
 - Biggest usage during conversion, but very stable
 - Hardly a blip (if any) for analysis
 - Limiting resource is your patience
- Much faster than using dataframes
 - even for smaller attack traces of tens of megabytes
- ddos_dissector adopted duckdb approach in favour of dataframes