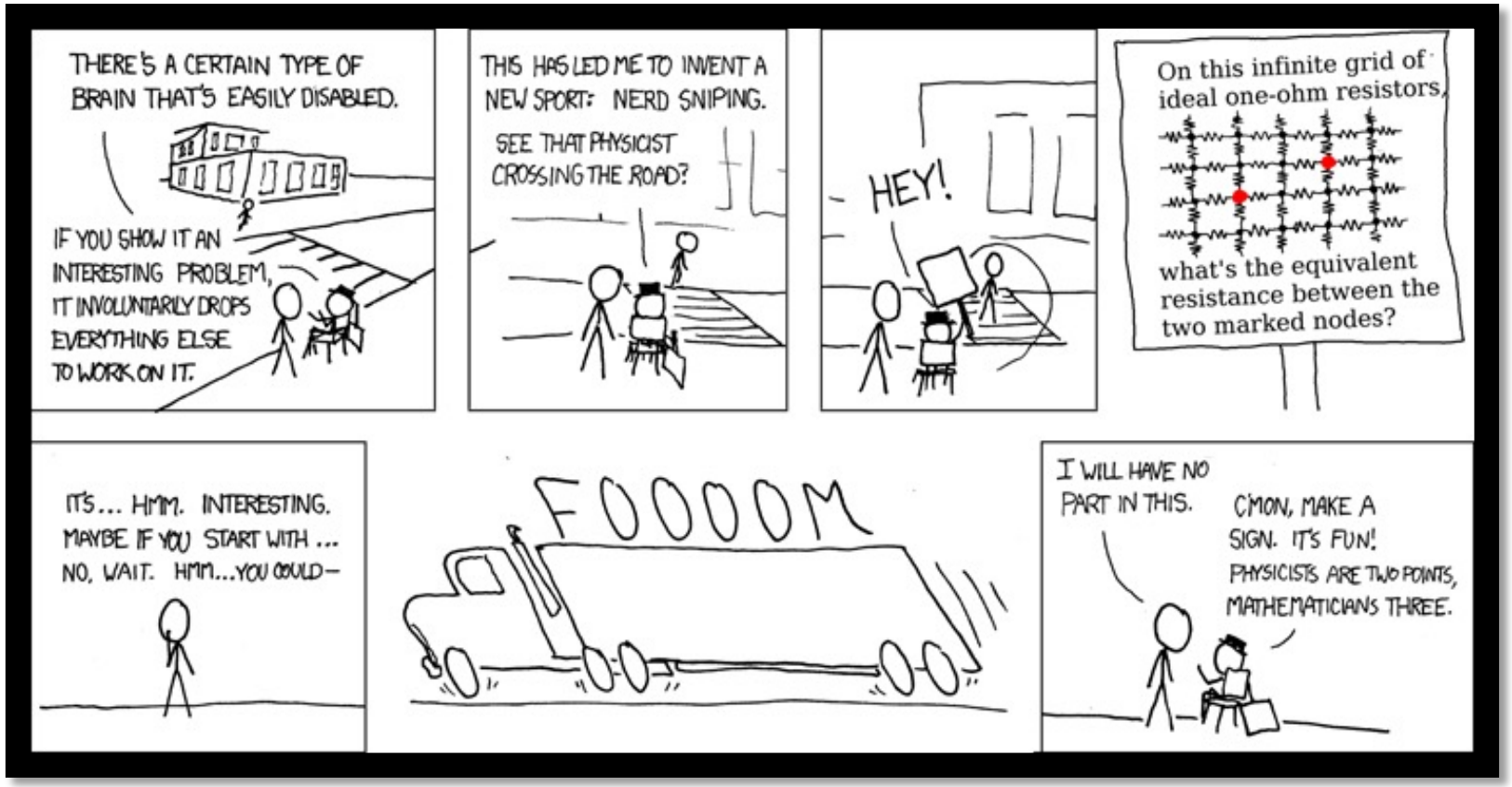
Speeding up Python Code

Josh Clark


THERE'S A CERTAIN TYPE OF BRAIN THAT'S EASILY DISABLED.
IF YOU SHOW IT AN INTERESTING PROBLEM, IT INVOLUNTARILY DROPS EVERYTHING ELSE TO WORK ON IT.
THIS HAS LED ME TO INVENT A NEW SPORT: NERD SNIPING.
SEE THAT PHYSICIST CROSSING THE ROAD?
HEY!
On this infinite grid of ideal one-ohm resistors,
what's the equivalent resistance between the two marked nodes?
IT'S... HMM. INTERESTING. MAYBE IF YOU START WITH ... NO, WAIT. HMM...YOU COULD—
FOOOOM
I WILL HAVE NO PART IN THIS.
C'MON, MAKE A SIGN. IT'S FUN! PHYSICISTS ARE TWO POINTS, MATHEMATICIANS THREE.

I Got Nerd Sniped

[Wireshark-dev] 90GB pcap file get last frame time stamp

Raj sekar mrajsekar@gmail.com via wireshark.org Fri, Feb 26, 12:09 PM
to Developer

Hi Everyone!

Need a help. Is there any library or method to get large pcap file's(offline) last timestamp.

I know capinfos can get this. But i want faster than capinfos.

Any suggestion?

Thanks

Best Regards
Raj

Sent via: Wireshark-dev mailing list <wireshark-dev@wireshark.org>
Archives: https://www.wireshark.org/lists/wireshark-dev
Unsubscribe: https://www.wireshark.org/mailman/options/wireshark-dev
mailto:wireshark-dev-request@wireshark.org?subject=unsubscribe

Capinfos

```
josh@joshs-mbp last_packet % capinfos maccdc2011_00013_20110312202724.pcap
File name:           maccdc2011_00013_20110312202724.pcap
File type:           Wireshark/tcpdump/... - pcap
File encapsulation:  Ethernet
File timestamp precision:  microseconds (6)
Packet size limit:   file hdr: 4096 bytes
Number of packets:   10M
File size:           4,430MB
Data size:           4,270MB
Capture duration:    11147.170981 seconds
First packet time:   2011-03-12 14:27:24.118438
Last packet time:    2011-03-12 17:33:11.289419
Data byte rate:      383kBps
Data bit rate:       3,064kbps
Average packet size: 427.06 bytes
Average packet rate: 897 packets/s
SHA256:              acbbfca5f88c95605af7f6a7029472614a29b5303d9e7d57167fd1268286d639
RIPEMD160:           c7a0e5e27d576eb78fb88968893f9d64e24365ea
SHA1:                f17959851aef35819d0efe269d156c19d7faa45c
Strict time order:   True
Number of interfaces in file: 1
Interface #0 info:
                     Encapsulation = Ethernet (1 - ether)
                     Capture length = 4096
                     Time precision = microseconds (6)
                     Time ticks per second = 1000000
                     Number of stat entries = 0
                     Number of packets = 10000000
```

What Does PCAP Look Like?

Global Header	Packet Header	Packet Data	Packet Header	Packet Data	Packet Header	Packet Data	...

```
typedef struct pcap_hdr_s {
        guint32 magic_number;   /* magic number */
        guint16 version_major;  /* major version number */
        guint16 version_minor;  /* minor version number */
        gint32  thiszone;       /* GMT to local correction */
        guint32 sigfigs;        /* accuracy of timestamps */
        guint32 snaplen;        /* max length of captured packets, in octets */
        guint32 network;        /* data link type */
} pcap_hdr_t;
```

```
typedef struct pcaprec_hdr_s {
        guint32 ts_sec;         /* timestamp seconds */
        guint32 ts_usec;        /* timestamp microseconds */
        guint32 incl_len;       /* number of octets of packet saved in file */
        guint32 orig_len;       /* actual length of packet */
} pcaprec_hdr_t;
```

What Does Our Code Have to Do?

1. Check the magic number to get the endianness of the file
2. For each packet header:
    1. Read and interpret the incl_len value
    2. Use that to skip ahead to the next packet header
3. When there are no more packets
    1. Get the ts_sec value from the last packet header
    2. Interpret that value to a readable timestamp
    3. Print the timestamp

Initial Performance for Comparison

```
josh@joshs-mbp last_packet % time capinfos maccdc2011_00013_20110312202724.pcap
File name:           maccdc2011_00013_20110312202724.pcap
File type:           Wireshark/tcpdump/... - pcap
File encapsulation:  Ethernet
File timestamp precision:  microseconds (6)
Packet size limit:   file hdr: 4096 bytes
Number of packets:   10M
File size:           4,430MB
Data size:           4,270MB
Capture duration:    11147.170981 seconds
First packet time:   2011-03-12 14:27:24.118438
Last packet time:    2011-03-12 17:33:11.289419
Data byte rate:      383kBps
Data bit rate:       3,064kbps
Average packet size: 427.06 bytes
Average packet rate: 897 packets/s
SHA256:              acbbfca5f88c95605af7f6a7029472614a29b5303d9e7d57167fd1268286d639
RIPEMD160:           c7a0e5e27d576eb78fb88968893f9d64e24365ea
SHA1:                f17959851aef35819d0efe269d156c19d7faa45c
Strict time order:   True
Number of interfaces in file: 1
Interface #0 info:
                     Encapsulation = Ethernet (1 - ether)
                     Capture length = 4096
                     Time precision = microseconds (6)
                     Time ticks per second = 1000000
                     Number of stat entries = 0
                     Number of packets = 10000000
capinfos maccdc2011_00013_20110312202724.pcap  47.28s user 3.01s system 95% cpu 52.510 total
```

Attempt 0 – Doing things the easiest way

```
time python3 last_packet_attempt_0.py maccdc2011_00013_20110312202724.pcap
This file was generated on a little-endian computer
first timestamp is 2011-03-12 14:27:24
Last packet's timestamp is 2011-03-12 17:33:11
python3 last_packet_attempt_0.py maccdc2011_00013_20110312202724.pcap  12.34s user 2.11s system 96% cpu 14.973 total
```

Attempt 1 - Convert bytes to int sparingly

```
josh@joshs-mbp last_packet % time python3 last_packet_attempt_1.py maccdc2011_00013_20110312202724.pcap
This file was generated on a little-endian computer
first timestamp is 2011-03-12 14:27:24
Last packet's timestamp is 2011-03-12 17:33:11
python3 last_packet_attempt_1.py maccdc2011_00013_20110312202724.pcap  9.94s user 2.20s system 94% cpu 12.807 total
```

cProfile – Let’s Get Analytical!

```
josh@joshs-mbp last_packet % python3 -m cProfile -s tottime last_packet_attempt_1.py maccdc2011_00013_20110312202724.pcap
This file was generated on a little-endian computer
first timestamp is 2011-03-12 14:27:24
Last packet's timestamp is 2011-03-12 17:33:11
         30005486 function calls (30005379 primitive calls) in 17.828 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    8.922    8.922   17.810   17.810 last_packet_attempt_1.py:20(last_timestamp)
 10000002    4.222    0.000    4.222    0.000 {method 'read' of '_io.BufferedRandom' objects}
 10000000    3.012    0.000    3.012    0.000 {method 'seek' of '_io.BufferedRandom' objects}
 10000026    1.654    0.000    1.654    0.000 {built-in method from_bytes}
        6    0.003    0.000    0.003    0.000 {built-in method _imp.create_dynamic}
        8    0.002    0.000    0.002    0.000 {built-in method marshal.loads}
        8    0.002    0.000    0.002    0.000 {method 'read' of '_io.BufferedReader' objects}
       82    0.001    0.000    0.001    0.000 {built-in method posix.stat}
    58/56    0.001    0.000    0.001    0.000 {built-in method builtins.__build_class__}
       14    0.001    0.000    0.001    0.000 {built-in method posix.getcwd}
```

Hold on – why is there a difference between total time and cumulative time?

We can do better – Enter kernprof

1. Pip install line_profiler
2. Add the @profile decorator to a function in the Python script
3. Run ‘kernprof –l –v <script_name>’

Kernprof Output

```
josh@joshs-mbp last_packet % kernprof -l -v last_packet_attempt_1.py maccdc2011_00013_20110312202724.pcap
This file was generated on a little-endian computer
first timestamp is 2011-03-12 14:27:24
Last packet's timestamp is 2011-03-12 17:33:11
Wrote profile results to last_packet_attempt_1.py.lprof
Timer unit: 1e-06 s

Total time: 58.4868 s
File: last_packet_attempt_1.py
Function: last_timestamp at line 20

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    20                                           @profile
    21                                           def last_timestamp(infile): # Expecting most of this to be command line input, so strings
    22
    23                                               # Open input file
    24         1         83.0     83.0      0.0      in_ptr = open(infile, 'rb+')
    25
    26                                               # Global Header
    27         1         14.0     14.0      0.0      global_header = in_ptr.read(24)
    28         1          4.0      4.0      0.0      endianness = determine_endianness(global_header)
    29         1         52.0     52.0      0.0      print(f'This file was generated on a {endianness}-endian computer')
    30
    31         1          2.0      2.0      0.0      packet_header = in_ptr.read(16)
    32
    33         1          1.0      1.0      0.0      timestamp = packet_header[0:4]
    34         1          4.0      4.0      0.0      packet_length = int.from_bytes(packet_header[8:12], byteorder=endianness)
    35         1         36.0     36.0      0.0      print(f"first timestamp is {datetime.fromtimestamp(int.from_bytes(timestamp, byteorder=endianness))}")
    36
    37         1          2.0      2.0      0.0      in_ptr.seek(packet_length,1)
    38
    39                                               # Packets
    40                                               while True:
    41                                                   # Packet Header
    42  10000000   14969054.0      1.5     25.6          packet_header = in_ptr.read(16)
    43  10000000    7709892.0      0.8     13.2          if not packet_header:
    44         1          1.0      1.0      0.0              break # Python returns '' for EOF, which evaluates FALSE
    45
    46   9999999    9490305.0      0.9     16.2          timestamp = packet_header[0:4]
    47   9999999   13097790.0      1.3     22.4          packet_length = int.from_bytes(packet_header[8:12], byteorder=endianness)
    48                                                   #print(f"timestamp is {timestamp} - pkt len is {packet_length}")
    49
    50                                                   # Move the file pointer to the beginning of the
    51                                                   # next packet header
    52   9999999   13219479.0      1.3     22.6          in_ptr.seek(packet_length,1)
    53
    54                                               # Be nice to your pointers. Put them away when you're done with them
    55         1         48.0     48.0      0.0      print(f"Last packet's timestamp is {datetime.fromtimestamp(int.from_bytes(timestamp, byteorder=endianness))}")
    56         1         20.0     20.0      0.0      in_ptr.close()
    57
```


```
    41                                                   # Packet Header
    42  10000000   14969054.0      1.5     25.6          packet_header = in_ptr.read(16)
    43  10000000    7709892.0      0.8     13.2          if not packet_header:
    44         1          1.0      1.0      0.0              break # Python returns '' for EOF, which evaluates FALSE
    45
    46   9999999    9490305.0      0.9     16.2          timestamp = packet_header[0:4]
    47   9999999   13097790.0      1.3     22.4          packet_length = int.from_bytes(packet_header[8:12], byteorder=endianness)
    48                                                   #print(f"timestamp is {timestamp} - pkt len is {packet_length}")
    49
    50                                                   # Move the file pointer to the beginning of the
    51                                                   # next packet header
    52   9999999   13219479.0      1.3     22.6          in_ptr.seek(packet_length,1)
    53
```

Attempt 2 – Fewer File Operations

```
josh@joshs-mbp last_packet % time python3 last_packet_attempt_2.py maccdc2011_00013_20110312202724.pcap
This file was generated on a little-endian computer
first timestamp is 2011-03-12 14:27:24
Last packet's timestamp is 2011-03-12 17:33:11
python3 last_packet_attempt_2.py maccdc2011_00013_20110312202724.pcap  9.59s user 2.10s system 96% cpu 12.079 total
```

```
                                                # Packets
                                                while True:
                                                    # Packet Header
 10000000   15074538.0      1.5     35.2          packet_header = in_ptr.read(chunk_size := 16 + packet_length)
 10000000    8135635.0      0.8     19.0          if len(packet_header) != (chunk_size):
        1          0.0      0.0      0.0              break # Python returns '' for EOF, which evaluates FALSE

  9999999    8567339.0      0.9     20.0          timestamp = packet_header[-16:-12]
                                                  #packet_len_bytes = packet_header[-8:-4]
  9999999   11080288.0      1.1     25.9          packet_length = int.from_bytes(packet_header[-8:-4], byteorder=endianness)
```

Attempt 3 – Infinite Loop No Longer

```
josh@joshs-mbp last_packet % time python3 last_packet_attempt_3.py maccdc2011_00013_20110312202724.pcap
This file was generated on a little-endian computer
first timestamp is 2011-03-12 14:27:24
Last packet's timestamp is 2011-03-12 17:33:11
python3 last_packet_attempt_3.py maccdc2011_00013_20110312202724.pcap  9.39s user 2.01s system 97% cpu 11.642 total
```

```
40                                                  # Packets
41  10000001    7887986.0      0.8     19.2         while len(packet_header) == (chunk_size):
42
43  10000000    8361410.0      0.8     20.3             timestamp = packet_header[-16:-12]
44  10000000   10752107.0      1.1     26.1             packet_length = int.from_bytes(packet_header[-8:-4], byteorder=endianness)
45
46  10000000   14131653.0      1.4     34.4             packet_header = in_ptr.read(chunk_size := 16 + packet_length)
```

Attempt 4 – Only Read Timestamp Once

```
[josh@joshs-mbp last_packet % time python3 last_packet_attempt_4.py maccdc2011_00013_20110312202724.pcap
This file was generated on a little-endian computer
first timestamp is 2011-03-12 14:27:24
Last packet's timestamp is 2011-03-12 17:33:11
python3 last_packet_attempt_4.py maccdc2011_00013_20110312202724.pcap  15.09s user 4.27s system 89% cpu 21.698 total
```

```
[josh@joshs-mbp last_packet % python3 -m cProfile -s cumtime last_packet_attempt_4.py maccdc2011_00013_20110312202724.pcap
This file was generated on a little-endian computer
first timestamp is 2011-03-12 14:27:24
Last packet's timestamp is 2011-03-12 17:33:11
         30005486 function calls (30005379 primitive calls) in 24.063 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      9/1    0.000    0.000   24.063   24.063 {built-in method builtins.exec}
        1    0.000    0.000   24.063   24.063 last_packet_attempt_4.py:1(<module>)
        1    9.303    9.303   24.043   24.043 last_packet_attempt_4.py:21(last_timestamp)
 10000000    7.880    0.000    7.880    0.000 {method 'tell' of '_io.BufferedRandom' objects}
 10000001    5.281    0.000    5.281    0.000 {method 'read' of '_io.BufferedRandom' objects}
```

Why is tell() so slow?

```
def tell(self):
    """Return an int indicating the current stream position."""
    return self.seek(0, 1)
```

What did we learn?

- When doing things a lot of times, it’s worth spending some time economizing
- cProfile is a great, built-in way to analyze time spent in functions/methods
- Kernprof (line_profiler) is a great way to analyze time spent on individual lines
- f.tell() is just f.seek() in disguise!
- I am very susceptible to nerd sniping