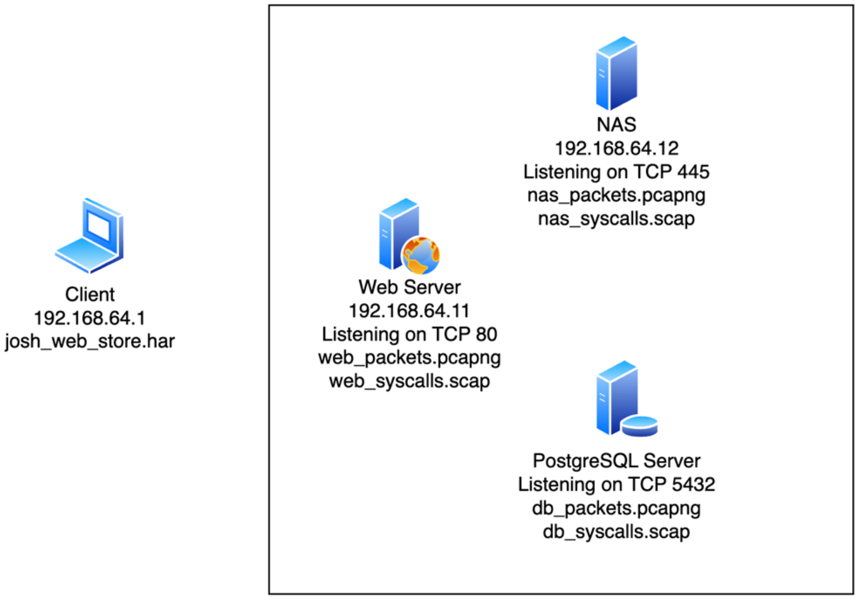
Distributed Application Troubleshooting with Wireshark and Stratoshark

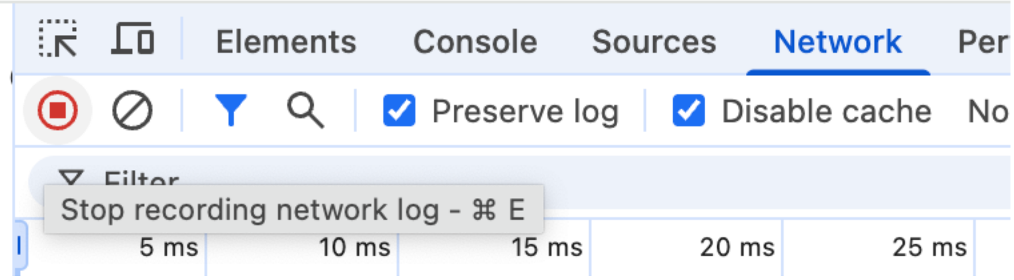
This exercise uses the files in the distributed_application_captures folders to troubleshoot a distributed web application. For an extra challenge, use only the first page to complete the exercise.



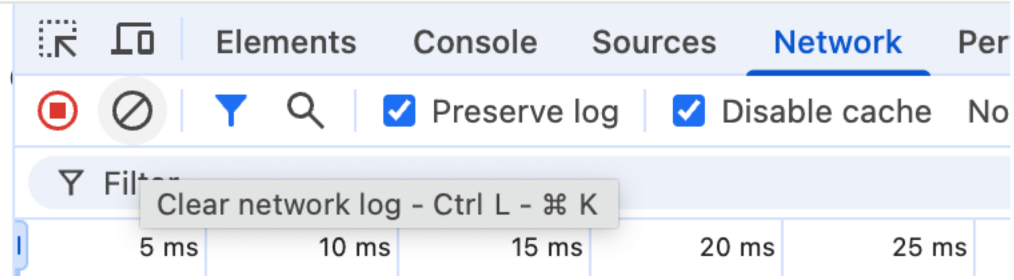
Use josh_web_store.har to identify the 3 slowest calls, and then use the included packet and system call captures to identify why each call is slow.

1. desk-chair.jpg - backend NAS call required with 30ms latency to the NAS
   - this SMB2 connection is single-threaded, so there is head-of-line blocking when the client requests all the product images at once
   - Solutions: move the NAS closer to the web server, implement caching on the web server, enable SMB multi-threading
2. products - backend database call with <1 ms network latency
   - PostgreSQL logs that the query itself took 3 sec
   - There are epoll_pwait() system calls that indicate the database pausing itself
   - Solutions: examine the stored procedure for query performance improvements
3. babel.min.js - backend NAS call required with 30ms latency to the NAS
   - This is a fairly large asset (3MB), so SMB2's low maximum read size requires several application turns to retrieve the entire file
   - Solutions: move the NAS closer to the web server, implement caching on the web server, increase the maximum read size in SMB2

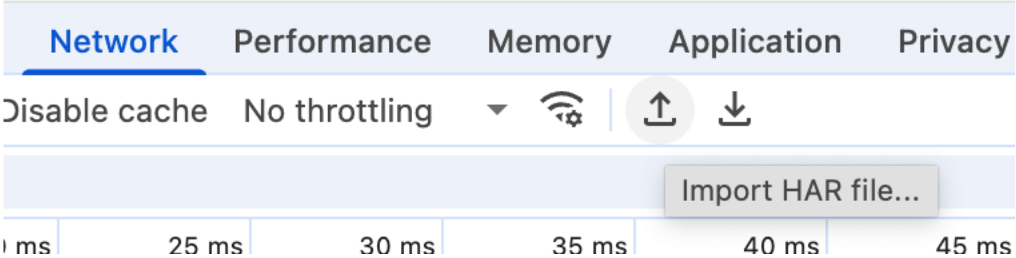
1. In your favorite web browser, open developer tools.
    a. In Chrome, you can go to View -> Developer -> Developer Tools. Click on the Network tab and click the “Stop Recording Network Log” Button



    b. If there are any network calls in the list, clear them



    c. Click the Up Arrow to import the file josh_web_store.har



    d. Click on the Time column header to sort by time. List out the 3 slowest calls, including their name and their overall load time

desk-chair.jpg - 3.96 sec
products - 3.07 sec
babel.min.js - 2.99 sec

2. Open web_packets.pcapng. We want to look at the packets to see where the latency occurs
    a. Filter to HTTP traffic
    b. Craft a filter to look for the HTTP request of the slowest call

http.request.uri contains "desk-chair.jpg"

c. Filter to that TCP session. What is the frame number of the last frame of the HTTP request?

2193

d. What is the frame number of the TCP acknowledgement of that request?

2197

e. What is the frame number of the first frame of the response data?

2449

f. The time difference between the TCP acknowledgement and the first frame of the response data is application delay. What is the application delay of this call?

3.95 seconds

g. Change your display filter to look at all traffic that occurs between the acknowledgement and the first frame of the response data using the format ‘frame.number in {<first_frame>..<second_frame>}’

frame.number in {2197..2449}

h. Looking at this traffic, does it appear that this call relies on backend calls to either the NAS or the database?

Yes, to the NAS

i. Repeat this process for the second and third slowest calls

products:

- http.request.uri contains "products"
- 2108
- 2109
- 2167
- frame.number in {2109..2167}
- Yes, to the database

babel.min.js:

- http.request.uri contains "babel.min.js"
- 28
- 30
- 239
- frame.number in {30..239}
- Yes, to the NAS

3. Troubleshooting NAS calls: use the following process to troubleshoot slow calls that rely on the NAS

    a. Create a filter that allows you to look at the HTTP request frames and the SMB2 traffic

http.request.uri contains "desk-chair.jpg" or smb2

    b. How long after the HTTP request does the web server send an SMB2 Open File request for the requested file?

3.7 seconds

    c. How long does the web server take to download the file from the NAS? This will be the time between the Create Request File and the last Read Response for that file.

243 milliseconds

    d. Once the file is received, how long does it take the web server to begin sending it back to the client. This will be the time difference between the last Read Response and the first frame of the HTTP response.

775 microseconds

    e. Where is the bulk of the slowdown?

Opening the file on the SMB2 share

f. Look closely at the traffic for that portion of the backend call. What could be causing the slowdown?

There are lots of files being requested at once, and only one SMB2 connection

g. If there are other slow NAS calls, repeat the troubleshooting process here.

babel.min.js:

- http.request.uri contains "babel.min.js" or smb2
- 1.02 seconds
- 1.95 seconds
- 35 milliseconds
- split between opening the file and downloading the file from the NAS to the web server
- the 1.02 second delay is the same single-threaded head of line blocking as desk-chair.jpg, but the 1.95 second delay happens because SMB2 enforces a limited maximum read size. There are several application turns required to get the full file.

4. Troubleshoot DB calls: use the following process to troubleshoot slow calls that rely on the database
    a. Open the db_syscalls.scap file

    b. Create a display filter to find network interactions with the web server using the format ‘fd.name contains “<web_server_IP>”

fd.name contains "192.168.64.11"

    c. Which event contains the backend request?

32034

    d. What is the SQL query?

SELECT * FROM get_products();

    e. Which event contains the response to the web server?

42189

    f. Set up the Find tool to look for Event Details and search for that SQL query. Where else does it show up?

log entries to /var/log/postgresql/postgresql-16-main.log

    g. Databases often record query performance. Can you find the query performance message for this query?

duration: 2929.134 ms

    h. Between the request and response, does postgres make any system calls that look like they could contribute to query latency? Look at the “Latency (s)” column.

There's 2 seconds of latency coming back from epoll_pwait()

i. If there are any other slow database calls, repeat the process here

N/As