PyPy vs CPython — VPN Benchmarking Guide

This document explains how to run the VPN project under CPython and PyPy, collect numeric benchmark results, and compare them. It includes the essential commands and example screenshots (where available).

Prerequisites

Host: Ubuntu 22.04 LTS (Jammy). Ensure you have Docker and docker-compose (or Docker Compose plugin) installed and working. The repo root should be at ~/vpnproject/src.

Open three terminal windows or panes and label them:

Terminal A — Host / Control (docker-compose, run benchmark scripts)

Terminal B — Router container (server logs and debug)

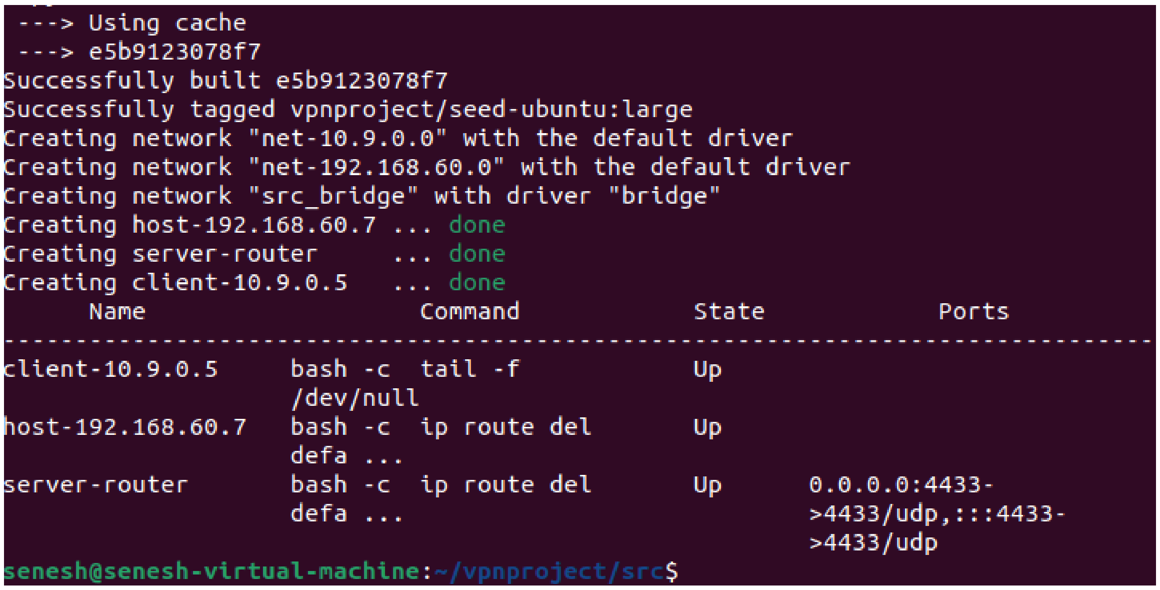
Terminal C — VPN_Client container (client logs, tun checks, ping)

Host: Install Docker & Setup

Run on Host (Terminal A):

```
sudo apt update
sudo apt install -y ca-certificates curl gnupg lsb-release
# Add Docker repo and install (official instructions)
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |   sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin
sudo usermod -aG docker $USER
newgrp docker
docker version
docker compose version
```

Example: after 'docker-compose up -d' the containers should be 'Up' (screenshot).



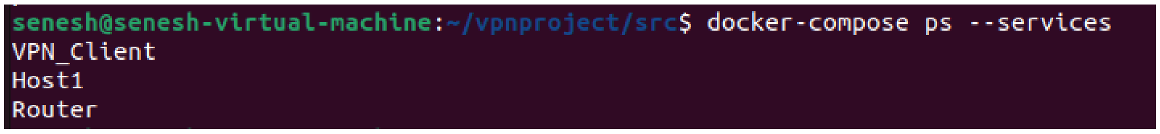
```
 ---> Using cache
 ---> e5b9123078f7
Successfully built e5b9123078f7
Successfully tagged vpnproject/seed-ubuntu:large
Creating network "net-10.9.0.0" with the default driver
Creating network "net-192.168.60.0" with the default driver
Creating network "src_bridge" with driver "bridge"
Creating host-192.168.60.7 ... done
Creating server-router     ... done
Creating client-10.9.0.5   ... done
      Name                   Command               State           Ports
-------------------------------------------------------------------------------
client-10.9.0.5     bash -c  tail -f           Up
                    /dev/null
host-192.168.60.7   bash -c  ip route del      Up
                    defa ...
server-router       bash -c  ip route del      Up      0.0.0.0:4433-
                    defa ...                           >4433/udp,:::4433-
                                                       >4433/udp
senesh@senesh-virtual-machine:~/vpnproject/src$
```

Caption: Containers are up and running.

Bring up the project containers

From Terminal A (Host), in the repo src folder run:

```
cd ~/vpnproject/src
docker-compose down -v
docker system prune -f
docker-compose build --no-cache
docker-compose up -d
docker-compose ps
```



```
senesh@senesh-virtual-machine:~/vpnproject/src$ docker-compose ps --services
VPN_Client
Host1
Router
```

Caption: docker-compose ps output showing service names and Up state.

Install Python interpreters and dependencies inside containers

Run these from Terminal A (Host) — they execute inside the containers with docker-compose exec. This installs PyPy and common dependencies for both Router and VPN_Client.

```
# Router
docker-compose exec Router bash -lc 'export DEBIAN_FRONTEND=noninteractive; apt-get
update -y; apt-get install -y pypy3 pypy3-dev python3-pip build-essential libssl-dev
libffi-dev iproute2 iptables tcpdump; python3 -m pip install -U pip; pypy3 -m pip
install -U pip'

# VPN_Client
```

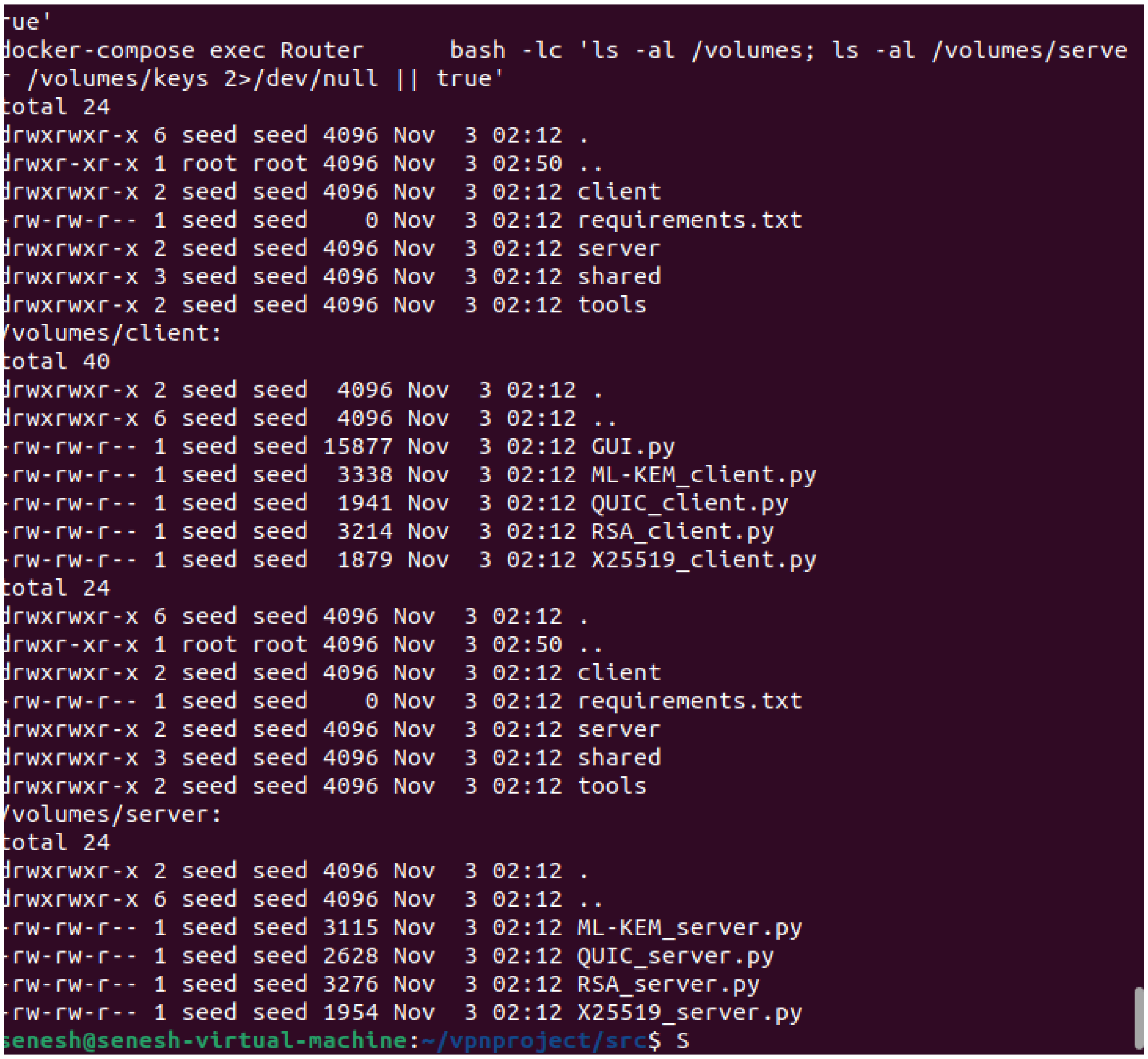
```
docker-compose exec VPN_Client bash -lc 'export DEBIAN_FRONTEND=noninteractive; apt-get
update -y; apt-get install -y pypy3 pypy3-dev python3-pip build-essential libssl-dev
libffi-dev iproute2 iptables tcpdump; python3 -m pip install -U pip; pypy3 -m pip
install -U pip'
```

Note: Installing some crypto libraries under PyPy may require building from source. If pip install for pypy3 fails for a package, capture the error and refer to the README troubleshooting section.

Ensure keys are present (server & client)

Check keys mounted under /volumes or /keys inside each container. Example (run from Host):

```
docker-compose exec Router bash -lc 'ls -l /volumes/keys || ls -l /keys/RSA'
docker-compose exec VPN_Client bash -lc 'ls -l /volumes/keys || ls -l /keys/RSA'
```



```
ue'
ocker-compose exec Router      bash -lc 'ls -al /volumes; ls -al /volumes/serve
 /volumes/keys 2>/dev/null || true'
otal 24
rwxrwxr-x 6 seed seed 4096 Nov  3 02:12 .
rwxr-xr-x 1 root root 4096 Nov  3 02:50 ..
rwxrwxr-x 2 seed seed 4096 Nov  3 02:12 client
rw-rw-r-- 1 seed seed    0 Nov  3 02:12 requirements.txt
rwxrwxr-x 2 seed seed 4096 Nov  3 02:12 server
rwxrwxr-x 3 seed seed 4096 Nov  3 02:12 shared
rwxrwxr-x 2 seed seed 4096 Nov  3 02:12 tools
volumes/client:
otal 40
rwxrwxr-x 2 seed seed  4096 Nov  3 02:12 .
rwxrwxr-x 6 seed seed  4096 Nov  3 02:12 ..
rw-rw-r-- 1 seed seed 15877 Nov  3 02:12 GUI.py
rw-rw-r-- 1 seed seed  3338 Nov  3 02:12 ML-KEM_client.py
rw-rw-r-- 1 seed seed  1941 Nov  3 02:12 QUIC_client.py
rw-rw-r-- 1 seed seed  3214 Nov  3 02:12 RSA_client.py
rw-rw-r-- 1 seed seed  1879 Nov  3 02:12 X25519_client.py
otal 24
rwxrwxr-x 6 seed seed 4096 Nov  3 02:12 .
rwxr-xr-x 1 root root 4096 Nov  3 02:50 ..
rwxrwxr-x 2 seed seed 4096 Nov  3 02:12 client
rw-rw-r-- 1 seed seed    0 Nov  3 02:12 requirements.txt
rwxrwxr-x 2 seed seed 4096 Nov  3 02:12 server
rwxrwxr-x 3 seed seed 4096 Nov  3 02:12 shared
rwxrwxr-x 2 seed seed 4096 Nov  3 02:12 tools
volumes/server:
otal 24
rwxrwxr-x 2 seed seed 4096 Nov  3 02:12 .
rwxrwxr-x 6 seed seed 4096 Nov  3 02:12 ..
rw-rw-r-- 1 seed seed 3115 Nov  3 02:12 ML-KEM_server.py
rw-rw-r-- 1 seed seed 2628 Nov  3 02:12 QUIC_server.py
rw-rw-r-- 1 seed seed 3276 Nov  3 02:12 RSA_server.py
rw-rw-r-- 1 seed seed 1954 Nov  3 02:12 X25519_server.py
enesh@senesh-virtual-machine:~/vpnproject/src$ S
```

Caption: Mounted volumes and keys listing inside container.

If keys are missing, generate them on Router and copy the server public key to Client. Example generate (Router):

```
docker-compose exec Router bash -lc 'mkdir -p /keys/RSA; openssl genrsa -out
/keys/RSA/server_private.pem 2048; openssl rsa -in /keys/RSA/server_private.pem -
outform PEM -pubout -out /keys/RSA/server_public.pem'
```

Copy server_public.pem from Router to Client (Host):

```
CID_R=$(docker-compose ps -q Router); CID_C=$(docker-compose ps -q VPN_Client)
docker cp $CID_R:/keys/RSA/server_public.pem ./server_public.pem
docker cp ./server_public.pem $CID_C:/keys/RSA/server_public.pem
rm ./server_public.pem
```

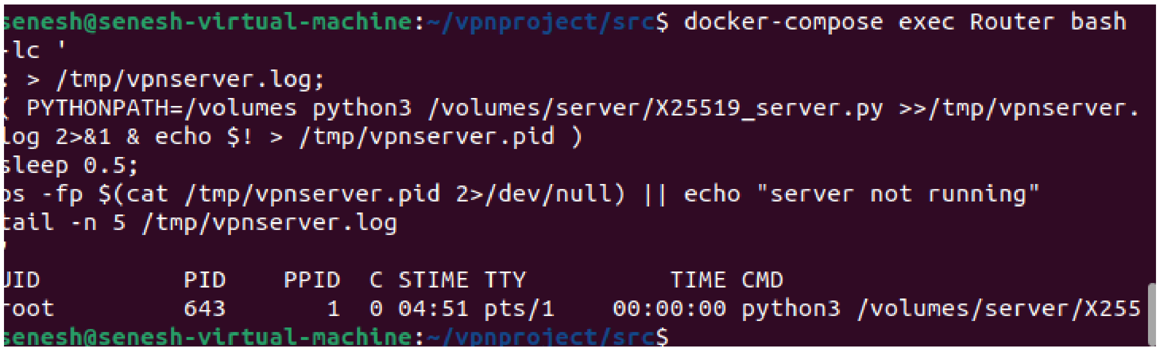
Run CPython server & client (sanity run)

In Terminal B (Router) start the server (background) and check listener:

```
:>/tmp/vpnserver.log; (PYTHONPATH=/volumes python3 -u /volumes/server/RSA_server.py
>>/tmp/vpnserver.log 2>&1 &); sleep 1; pgrep -af RSA_server.py || echo "server not
running"; ss -lunp | grep -E ":4433\s" || echo "no UDP:4433 listener"; tail -n 12
/tmp/vpnserver.log
```

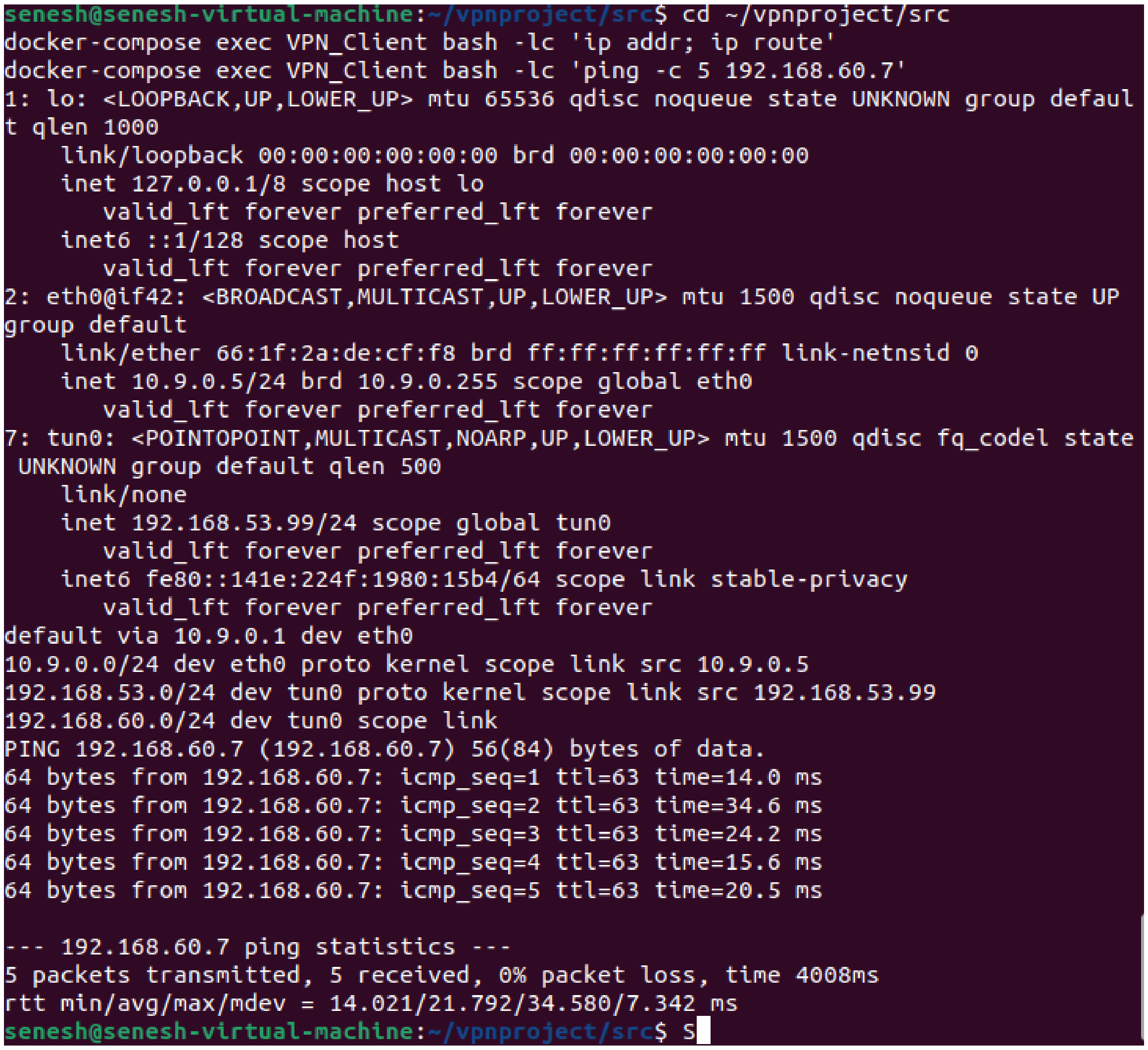
In Terminal C (VPN_Client) start the client and check tun0 and ping:

```
:>/tmp/vpnclient.log; (HOST=192.168.60.7 PORT=4433 PYTHONPATH=/volumes python3 -u
/volumes/client/RSA_client.py >>/tmp/vpnclient.log 2>&1 &); sleep 1; pgrep -af
RSA_client.py || echo "client not running"; ip addr show tun0 || echo "no tun0"; tail -
n 12 /tmp/vpnclient.log
docker-compose exec VPN_Client bash -lc 'ping -c 5 192.168.60.7 || true'
```



```
senesh@senesh-virtual-machine:~/vpnproject/src$ docker-compose exec Router bash
-lc '
: > /tmp/vpnserver.log;
( PYTHONPATH=/volumes python3 /volumes/server/X25519_server.py >>/tmp/vpnserver.
log 2>&1 & echo $! > /tmp/vpnserver.pid )
sleep 0.5;
ps -fp $(cat /tmp/vpnserver.pid 2>/dev/null) || echo "server not running"
tail -n 5 /tmp/vpnserver.log
'
UID          PID    PPID  C STIME TTY          TIME CMD
root         643       1  0 04:51 pts/1    00:00:00 python3 /volumes/server/X255
senesh@senesh-virtual-machine:~/vpnproject/src$
```

Caption: Server process running and tail of log (example).



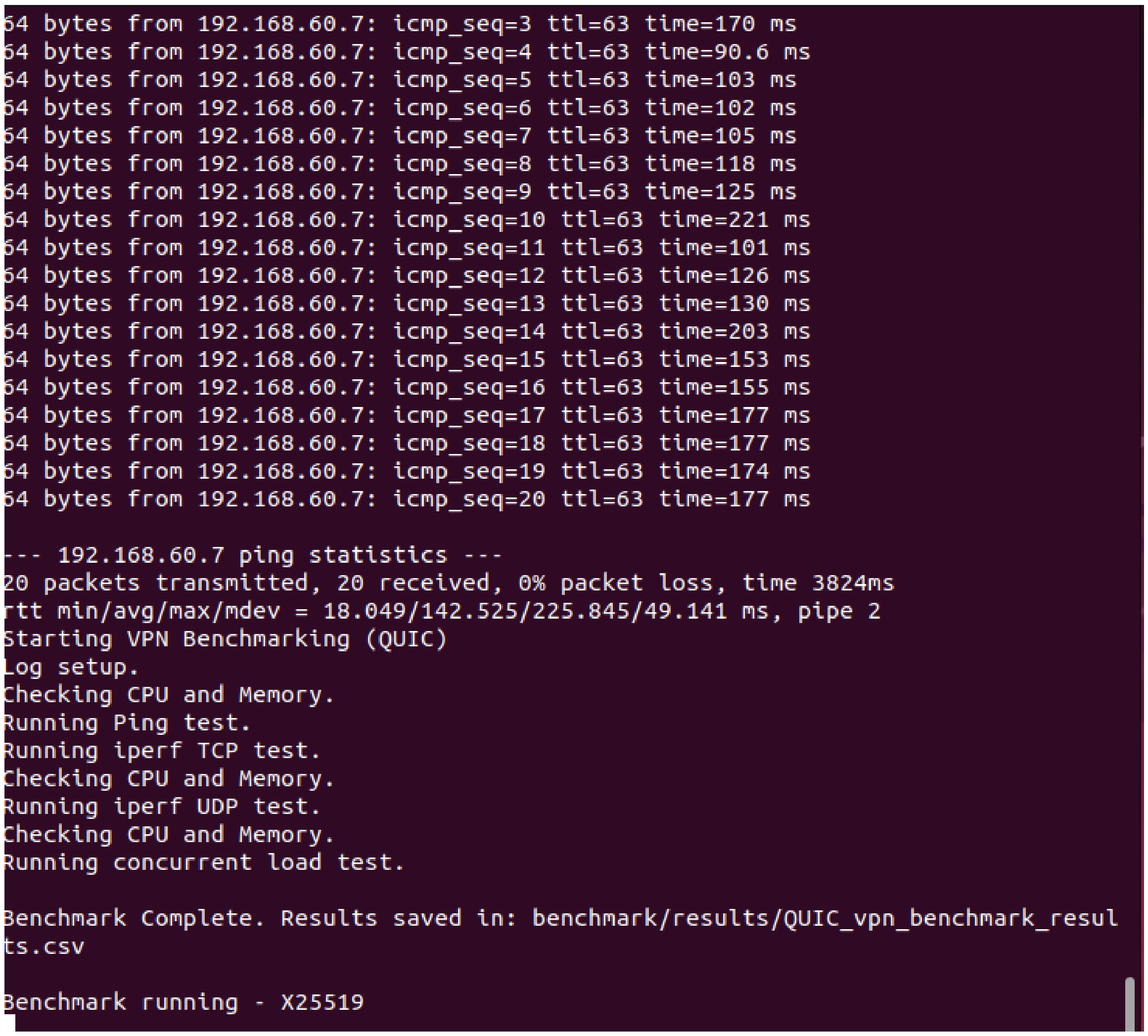
```
senesh@senesh-virtual-machine:~/vpnproject/src$ cd ~/vpnproject/src
docker-compose exec VPN_Client bash -lc 'ip addr; ip route'
docker-compose exec VPN_Client bash -lc 'ping -c 5 192.168.60.7'
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0@if42: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default
    link/ether 66:1f:2a:de:cf:f8 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
7: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
 UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::141e:224f:1980:15b4/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.53.0/24 dev tun0 proto kernel scope link src 192.168.53.99
192.168.60.0/24 dev tun0 scope link
PING 192.168.60.7 (192.168.60.7) 56(84) bytes of data.
64 bytes from 192.168.60.7: icmp_seq=1 ttl=63 time=14.0 ms
64 bytes from 192.168.60.7: icmp_seq=2 ttl=63 time=34.6 ms
64 bytes from 192.168.60.7: icmp_seq=3 ttl=63 time=24.2 ms
64 bytes from 192.168.60.7: icmp_seq=4 ttl=63 time=15.6 ms
64 bytes from 192.168.60.7: icmp_seq=5 ttl=63 time=20.5 ms

--- 192.168.60.7 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 14.021/21.792/34.580/7.342 ms
senesh@senesh-virtual-machine:~/vpnproject/src$ S
```

Caption: tun0 interface present and ping replies — tunnel established (example).

Run CPython benchmark

Run the benchmark harness from Host (Terminal A). Make sure server and client are running before starting.

```
# CPython baseline
cd ~/vpnproject/src
bash benchmark.sh --interpreter=python3 | tee ../benchmark_run_cpython.log
# results are saved under benchmark/results or ../results/cpython depending on script
settings
```



```
64 bytes from 192.168.60.7: icmp_seq=3 ttl=63 time=170 ms
64 bytes from 192.168.60.7: icmp_seq=4 ttl=63 time=90.6 ms
64 bytes from 192.168.60.7: icmp_seq=5 ttl=63 time=103 ms
64 bytes from 192.168.60.7: icmp_seq=6 ttl=63 time=102 ms
64 bytes from 192.168.60.7: icmp_seq=7 ttl=63 time=105 ms
64 bytes from 192.168.60.7: icmp_seq=8 ttl=63 time=118 ms
64 bytes from 192.168.60.7: icmp_seq=9 ttl=63 time=125 ms
64 bytes from 192.168.60.7: icmp_seq=10 ttl=63 time=221 ms
64 bytes from 192.168.60.7: icmp_seq=11 ttl=63 time=101 ms
64 bytes from 192.168.60.7: icmp_seq=12 ttl=63 time=126 ms
64 bytes from 192.168.60.7: icmp_seq=13 ttl=63 time=130 ms
64 bytes from 192.168.60.7: icmp_seq=14 ttl=63 time=203 ms
64 bytes from 192.168.60.7: icmp_seq=15 ttl=63 time=153 ms
64 bytes from 192.168.60.7: icmp_seq=16 ttl=63 time=155 ms
64 bytes from 192.168.60.7: icmp_seq=17 ttl=63 time=177 ms
64 bytes from 192.168.60.7: icmp_seq=18 ttl=63 time=177 ms
64 bytes from 192.168.60.7: icmp_seq=19 ttl=63 time=174 ms
64 bytes from 192.168.60.7: icmp_seq=20 ttl=63 time=177 ms

--- 192.168.60.7 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 3824ms
rtt min/avg/max/mdev = 18.049/142.525/225.845/49.141 ms, pipe 2
Starting VPN Benchmarking (QUIC)
Log setup.
Checking CPU and Memory.
Running Ping test.
Running iperf TCP test.
Checking CPU and Memory.
Running iperf UDP test.
Checking CPU and Memory.
Running concurrent load test.

Benchmark Complete. Results saved in: benchmark/results/QUIC_vpn_benchmark_resul
ts.csv

Benchmark running - X25519
```

Caption: Benchmark run completed and results saved (example).

Stop CPython processes and start PyPy versions

Stop CPython server & client (Terminal B and C):

```
# stop server (Router)
docker-compose exec Router bash -lc 'pkill -f RSA_server.py || true; sleep 0.5; pgrep -
af RSA_server.py || echo "server stopped"'
# stop client (VPN_Client)
docker-compose exec VPN_Client bash -lc 'pkill -f RSA_client.py || true; sleep 0.5;
pgrep -af RSA_client.py || echo "client stopped"'
```

Start server and client under PyPy (Terminal B and C):

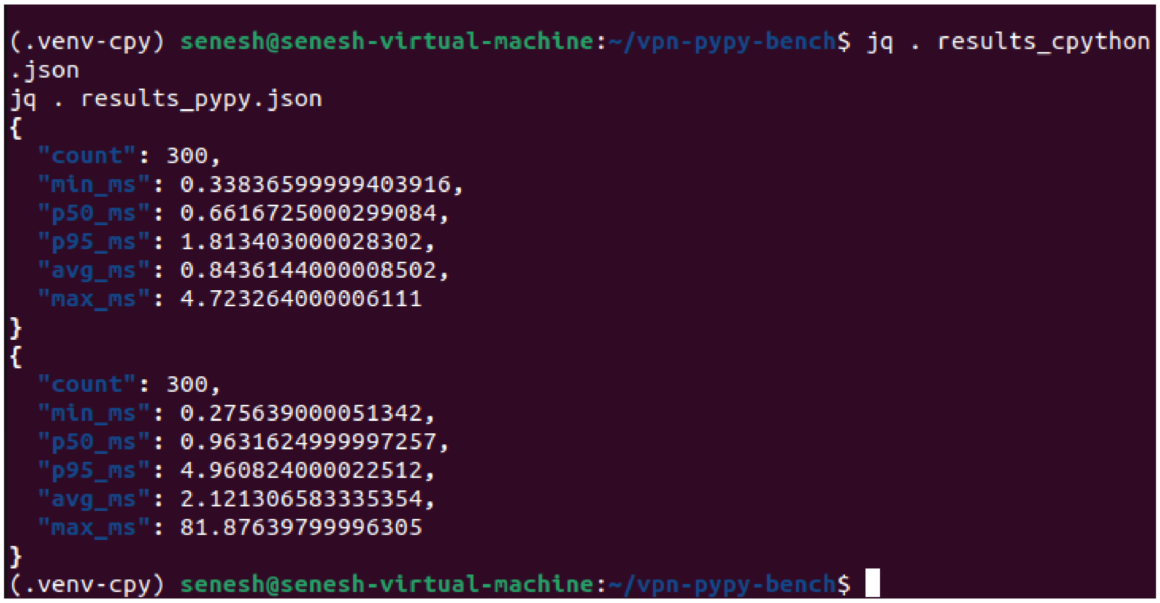
```
:>/tmp/vpnserver_pypy.log; (PYTHONPATH=/volumes pypy3 -u /volumes/server/RSA_server.py
>>/tmp/vpnserver_pypy.log 2>&1 &); sleep 1; pgrep -af RSA_server.py || echo "server not
running"; ss -lunp | grep -E ":4433\s" || echo "no UDP:4433 listener"
:>/tmp/vpnclient_pypy.log; (HOST=192.168.60.7 PORT=4433 PYTHONPATH=/volumes pypy3 -u
/volumes/client/RSA_client.py >>/tmp/vpnclient_pypy.log 2>&1 &); sleep 1; pgrep -af
```

```
RSA_client.py || echo "client not running"; ip addr show tun0 || echo "no tun0"; tail -
n 12 /tmp/vpnclient_pypy.log
```

Run PyPy benchmark

From Host (Terminal A) run benchmark with interpreter=pypy3:

```
cd ~/vpnproject/src
bash benchmark.sh --interpreter=pypy3 | tee ../benchmark_run_pypy.log
# copy results to ../results/pypy or as configured in benchmark.sh
```



```
(.venv-cpy) senesh@senesh-virtual-machine:~/vpn-pypy-bench$ jq . results_cpython
.json
jq . results_pypy.json
{
  "count": 300,
  "min_ms": 0.33836599999403916,
  "p50_ms": 0.6616725000299084,
  "p95_ms": 1.813403000028302,
  "avg_ms": 0.8436144000008502,
  "max_ms": 4.723264000006111
}
{
  "count": 300,
  "min_ms": 0.275639000051342,
  "p50_ms": 0.9631624999997257,
  "p95_ms": 4.960824000022512,
  "avg_ms": 2.121306583335354,
  "max_ms": 81.87639799996305
}
(.venv-cpy) senesh@senesh-virtual-machine:~/vpn-pypy-bench$
```

Caption: Example microbenchmark JSON with p50/p95/avg fields (example).

Compare CPython vs PyPy results

Use the included comparison script to produce a short CSV summary of deltas (place this script at scripts/compare_results.py):

```
# from repo root (Host)
python3 scripts/compare_results.py --cpython-dir ./results/cpython --pypy-dir
./results/pypy > results/summary_table.csv
# view
cat results/summary_table.csv
```

Troubleshooting & Tips

- If you see 'Permission denied' when running docker commands, add your user to the docker group: `sudo usermod -aG docker $USER` then `newgrp docker`.

- If bind errors (address in use) appear, stop conflicting process or change port and retry. Use `ss -lunp | grep 4433` to find who is using the port.

- If PyPy pip install fails for a package (C extension), install build deps inside container: `apt-get install build-essential libssl-dev libffi-dev` and retry.

- If benchmark CSVs contain `NaN`, inspect `/tmp/vpnserver.log` and `/tmp/vpnclient.log` and the 'raw_output' column in the CSV to understand missing iperf outputs.

Appendix: Useful commands

```
# Show server logs
docker-compose exec Router bash -lc 'tail -n 100 /tmp/vpnserver.log'
# Show client logs
docker-compose exec VPN_Client bash -lc 'tail -n 100 /tmp/vpnclient.log'
# Check tun and routes
docker-compose exec VPN_Client bash -lc 'ip addr show tun0; ip route'
# Stop all
docker-compose down -v
```