Reproducing Network Packet Loss and Network Delay Experiments: One-Size-Fits-None (NSDI’25)

Network Delay

# Initial Setup with blockade.yaml

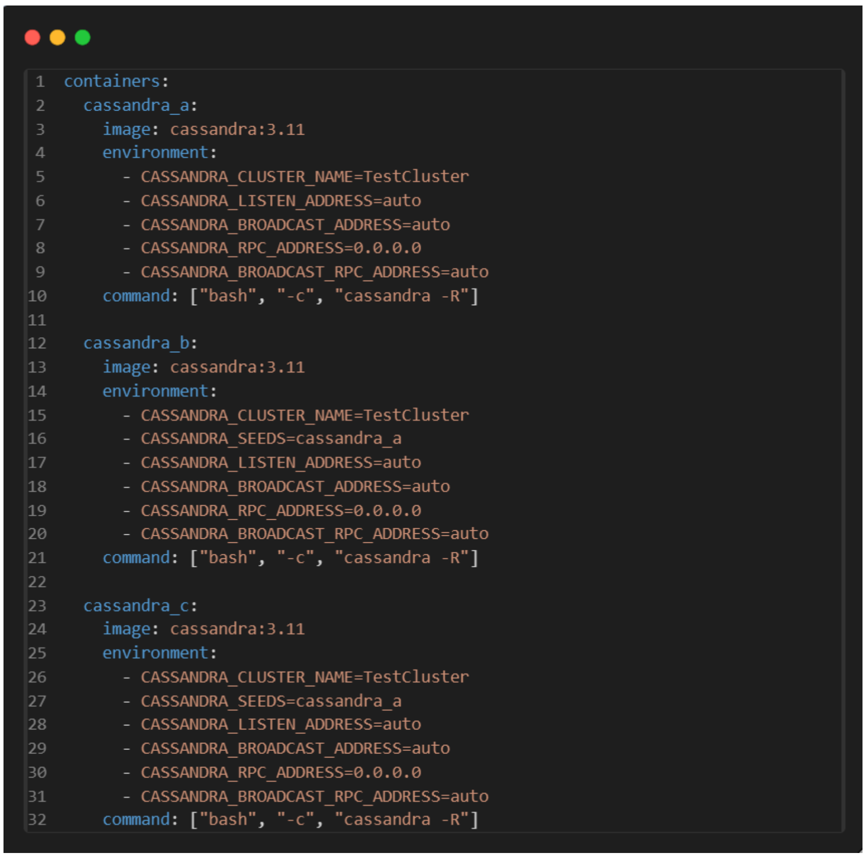
We started by trying to use Blockade as the delay injector, following the approach mentioned in the paper.

Problem Encountered:

Blockade by default creates its own Docker network, which was not aligned with our Docker Compose setup.

Fix:

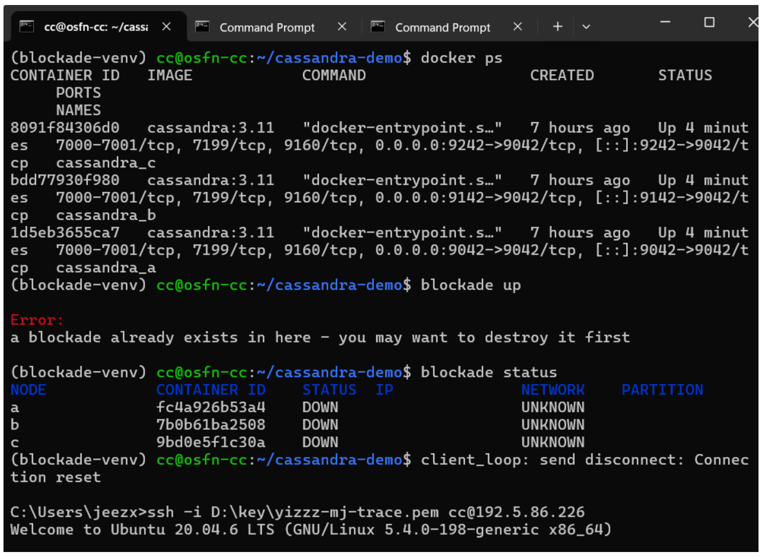
We modified the blockade.yaml file to match the network and container naming from docker-compose.yaml.



```yaml
containers:
  cassandra_a:
    image: cassandra:3.11
    environment:
      - CASSANDRA_CLUSTER_NAME=TestCluster
      - CASSANDRA_LISTEN_ADDRESS=auto
      - CASSANDRA_BROADCAST_ADDRESS=auto
      - CASSANDRA_RPC_ADDRESS=0.0.0.0
      - CASSANDRA_BROADCAST_RPC_ADDRESS=auto
    command: ["bash", "-c", "cassandra -R"]

  cassandra_b:
    image: cassandra:3.11
    environment:
      - CASSANDRA_CLUSTER_NAME=TestCluster
      - CASSANDRA_SEEDS=cassandra_a
      - CASSANDRA_LISTEN_ADDRESS=auto
      - CASSANDRA_BROADCAST_ADDRESS=auto
      - CASSANDRA_RPC_ADDRESS=0.0.0.0
      - CASSANDRA_BROADCAST_RPC_ADDRESS=auto
    command: ["bash", "-c", "cassandra -R"]

  cassandra_c:
    image: cassandra:3.11
    environment:
      - CASSANDRA_CLUSTER_NAME=TestCluster
      - CASSANDRA_SEEDS=cassandra_a
      - CASSANDRA_LISTEN_ADDRESS=auto
      - CASSANDRA_BROADCAST_ADDRESS=auto
      - CASSANDRA_RPC_ADDRESS=0.0.0.0
      - CASSANDRA_BROADCAST_RPC_ADDRESS=auto
    command: ["bash", "-c", "cassandra -R"]
```

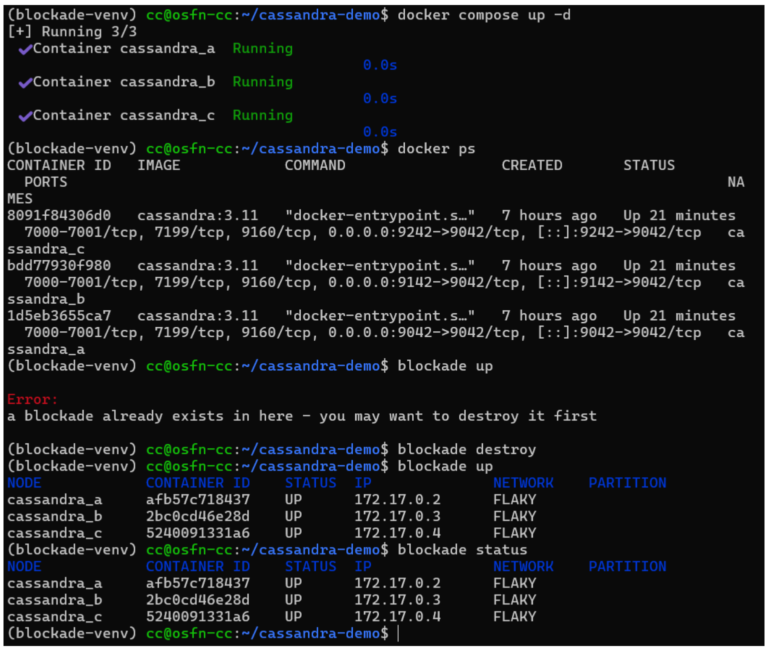
Also matched the network settings (bridge mode and subnet) accordingly.

Before



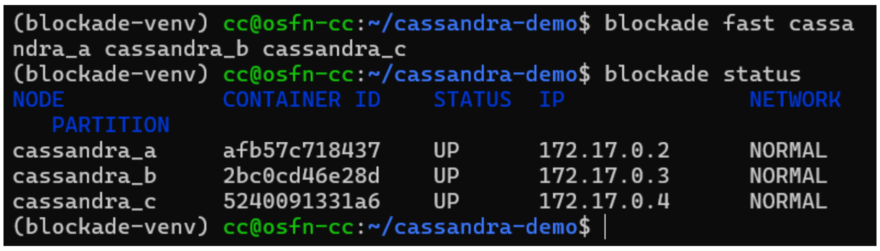
```
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED       STATUS
     PORTS
     NAMES
8091f84306d0   cassandra:3.11    "docker-entrypoint.s…"   7 hours ago   Up 4 minut
es   7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9242->9042/tcp, [::]:9242->9042/t
cp   cassandra_c
bdd77930f980   cassandra:3.11    "docker-entrypoint.s…"   7 hours ago   Up 4 minut
es   7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9142->9042/tcp, [::]:9142->9042/t
cp   cassandra_b
1d5eb3655ca7   cassandra:3.11    "docker-entrypoint.s…"   7 hours ago   Up 4 minut
es   7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp, [::]:9042->9042/t
cp   cassandra_a
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ blockade up

Error:
a blockade already exists in here - you may want to destroy it first

(blockade-venv) cc@osfn-cc:~/cassandra-demo$ blockade status
NODE            CONTAINER ID    STATUS  IP              NETWORK    PARTITION
a               fc4a926b53a4    DOWN                    UNKNOWN
b               7b0b61ba2508    DOWN                    UNKNOWN
c               9bd0e5f1c30a    DOWN                    UNKNOWN
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ client_loop: send disconnect: Connec
tion reset

C:\Users\jeezx>ssh -i D:\key\yizzz-mj-trace.pem cc@192.5.86.226
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-198-generic x86_64)
```

After



```
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ docker compose up -d
[+] Running 3/3
 ✔Container cassandra_a  Running
                                       0.0s
 ✔Container cassandra_b  Running
                                       0.0s
 ✔Container cassandra_c  Running
                                       0.0s
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ docker ps
CONTAINER ID   IMAGE            COMMAND                   CREATED       STATUS
 PORTS                                                                        NA
MES
8091f84306d0   cassandra:3.11   "docker-entrypoint.s…"   7 hours ago   Up 21 minutes
  7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9242->9042/tcp, [::]:9242->9042/tcp   ca
ssandra_c
bdd77930f980   cassandra:3.11   "docker-entrypoint.s…"   7 hours ago   Up 21 minutes
  7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9142->9042/tcp, [::]:9142->9042/tcp   ca
ssandra_b
1d5eb3655ca7   cassandra:3.11   "docker-entrypoint.s…"   7 hours ago   Up 21 minutes
  7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp, [::]:9042->9042/tcp   ca
ssandra_a
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ blockade up

Error:
a blockade already exists in here - you may want to destroy it first

(blockade-venv) cc@osfn-cc:~/cassandra-demo$ blockade destroy
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ blockade up
NODE            CONTAINER ID    STATUS  IP              NETWORK    PARTITION
cassandra_a     afb57c718437    UP      172.17.0.2      FLAKY
cassandra_b     2bc0cd46e28d    UP      172.17.0.3      FLAKY
cassandra_c     5240091331a6    UP      172.17.0.4      FLAKY
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ blockade status
NODE            CONTAINER ID    STATUS  IP              NETWORK    PARTITION
cassandra_a     afb57c718437    UP      172.17.0.2      FLAKY
cassandra_b     2bc0cd46e28d    UP      172.17.0.3      FLAKY
cassandra_c     5240091331a6    UP      172.17.0.4      FLAKY
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ |
```

# Reset to the fast mode



```
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ blockade fast cassa
ndra_a cassandra_b cassandra_c
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ blockade status
NODE            CONTAINER ID    STATUS  IP              NETWORK
   PARTITION
cassandra_a     afb57c718437    UP      172.17.0.2      NORMAL
cassandra_b     2bc0cd46e28d    UP      172.17.0.3      NORMAL
cassandra_c     5240091331a6    UP      172.17.0.4      NORMAL
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ |
```

We can see, the network status turn to NORMAL from the FLAKY status before

# Blockade Injection Test (Flaky / Packet Loss)

We attempted to inject packet loss:

blockade flaky cassandra_b 80%

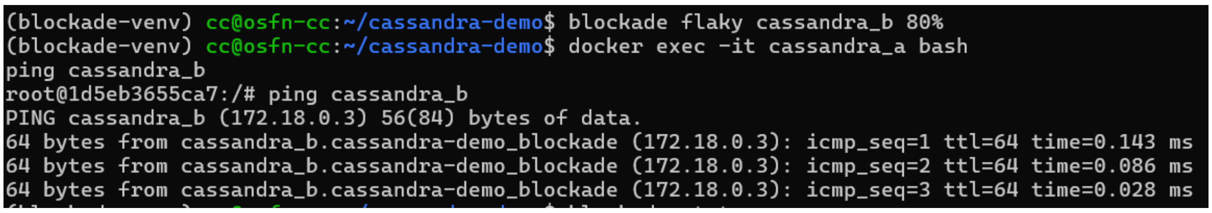
Then entered cassandra_a container:

docker exec -it cassandra_a bash
ping cassandra_b

Problem Encountered:

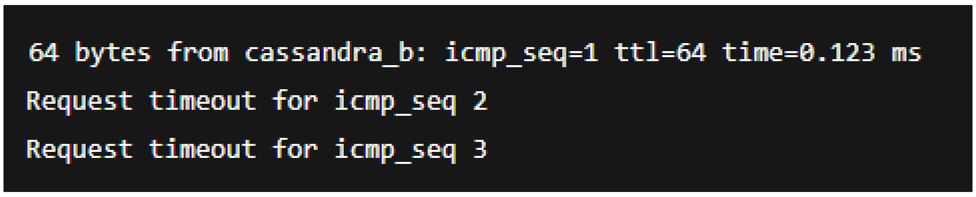
- There was no observable effect. Despite blockade status showing FLAKY, pings succeeded with no packet loss.

  blockade flaky cassandra_b 80%
  docker exec -it cassandra_a bash



```
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ blockade flaky cassandra_b 80%
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ docker exec -it cassandra_a bash
ping cassandra_b
root@1d5eb3655ca7:/# ping cassandra_b
PING cassandra_b (172.18.0.3) 56(84) bytes of data.
64 bytes from cassandra_b.cassandra-demo_blockade (172.18.0.3): icmp_seq=1 ttl=64 time=0.143 ms
64 bytes from cassandra_b.cassandra-demo_blockade (172.18.0.3): icmp_seq=2 ttl=64 time=0.086 ms
64 bytes from cassandra_b.cassandra-demo_blockade (172.18.0.3): icmp_seq=3 ttl=64 time=0.028 ms
```

We can see here it is not giving any effect. The expected result must be



```
64 bytes from cassandra_b: icmp_seq=1 ttl=64 time=0.123 ms
Request timeout for icmp_seq 2
Request timeout for icmp_seq 3
```

Despite multiple patching attempts, Blockade consistently failed to apply delays or packet loss effectively because of namespace and network isolation issues with Docker Compose-managed containers. Blockade is designed for containers it launches directly. Our Cassandra containers were launched via Docker Compose, and integrating these with Blockade's control flow proved to be brittle and inconsistent.

Therefore, we switched to using `tc netem` commands (via script) inside the containers to inject delay directly at the OS level.

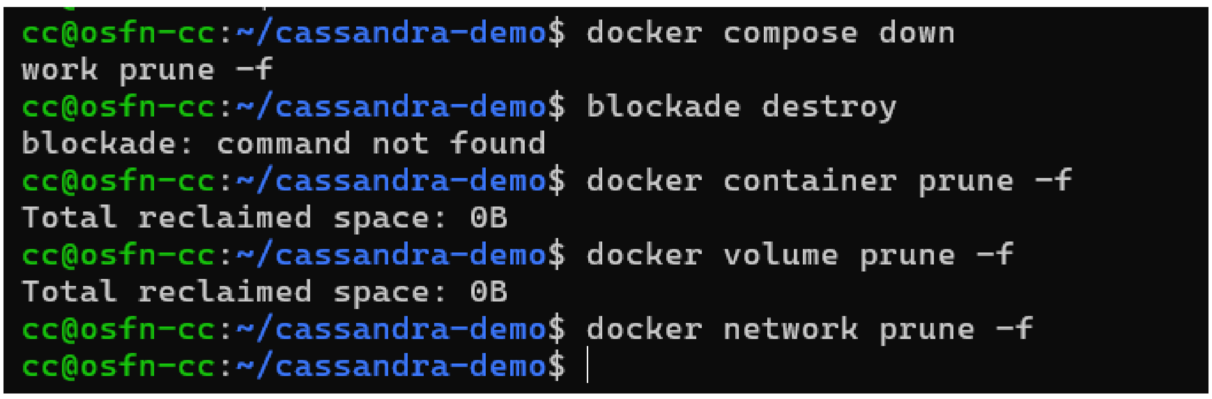
Hypothesis:

- Blockade's default bridge network was not correctly linked to Docker Compose's custom bridge.
- Containers launched by Docker Compose were not being effectively controlled by Blockade.

# Reset Network State (Cleanup)

To ensure no residual qdisc (traffic control) entries or dangling networks interfered:

docker compose down
blockade destroy
docker container prune -f
docker volume prune -f
docker network prune -f



```
cc@osfn-cc:~/cassandra-demo$ docker compose down
work prune -f
cc@osfn-cc:~/cassandra-demo$ blockade destroy
blockade: command not found
cc@osfn-cc:~/cassandra-demo$ docker container prune -f
Total reclaimed space: 0B
cc@osfn-cc:~/cassandra-demo$ docker volume prune -f
Total reclaimed space: 0B
cc@osfn-cc:~/cassandra-demo$ docker network prune -f
cc@osfn-cc:~/cassandra-demo$
```

We verified that containers were clean and rebuilt the network using only Docker Compose.

# Shift to Manual Injection using tc (run_netdelay.sh)

Due to Blockade's ineffectiveness, we developed a manual injection script using Linux tc (traffic control):

./run_netdelay.sh "cassandra_b cassandra_c" "100ms" 30 delay100ms_bc

This script:

1. Injects tc netem delay 100ms to cassandra_b and cassandra_c
2. Waits 15s for the cluster to stabilize
3. Runs cassandra-stress write on cassandra_a for 30s
4. Collects the result log and saves it to a unique folder
5. Removes the delay

# Verifying tc injection

Before each run, we check tc qdisc show:

docker exec cassandra_b tc qdisc show
docker exec cassandra_c tc qdisc show

It shows:

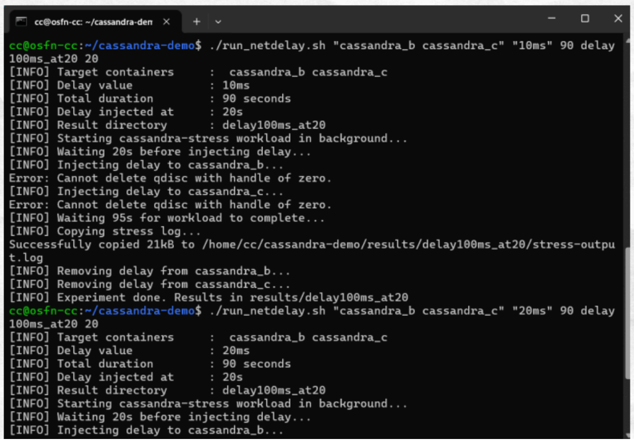
qdisc netem 803a: dev eth0 root refcnt 2 limit 1000 delay 100ms

This confirms the delay is correctly injected.

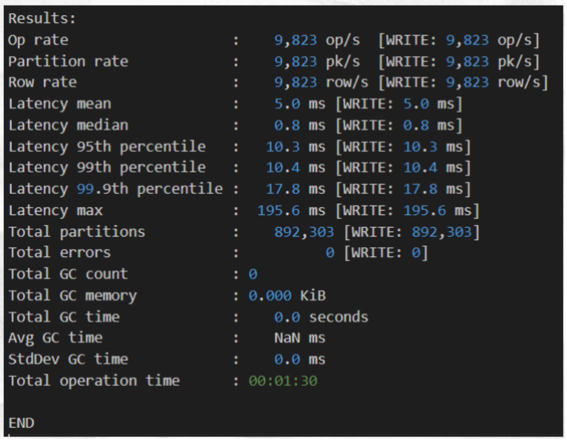
# Running Experiments

We successfully ran experiments with various delays:

- 1ms: baseline
- 50ms: intermediate
- 75ms: upper-intermediate
- 100ms: high delay



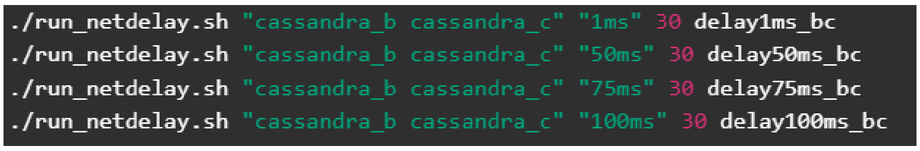
```
cc@osfn-cc:~/cassandra-demo$ ./run_netdelay.sh "cassandra_b cassandra_c" "10ms" 90 delay
100ms_at20 20
[INFO] Target containers     :  cassandra_b cassandra_c
[INFO] Delay value           : 10ms
[INFO] Total duration        : 90 seconds
[INFO] Delay injected at     : 20s
[INFO] Result directory      : delay100ms_at20
[INFO] Starting cassandra-stress workload in background...
[INFO] Waiting 20s before injecting delay...
[INFO] Injecting delay to cassandra_b...
Error: Cannot delete qdisc with handle of zero.
[INFO] Injecting delay to cassandra_c...
Error: Cannot delete qdisc with handle of zero.
[INFO] Waiting 95s for workload to complete...
[INFO] Copying stress log...
Successfully copied 21kB to /home/cc/cassandra-demo/results/delay100ms_at20/stress-outpu
t.log
[INFO] Removing delay from cassandra_b...
[INFO] Removing delay from cassandra_c...
[INFO] Experiment done. Results in results/delay100ms_at20
cc@osfn-cc:~/cassandra-demo$ ./run_netdelay.sh "cassandra_b cassandra_c" "20ms" 90 delay
100ms_at20 20
[INFO] Target containers     :  cassandra_b cassandra_c
[INFO] Delay value           : 20ms
[INFO] Total duration        : 90 seconds
[INFO] Delay injected at     : 20s
[INFO] Result directory      : delay100ms_at20
[INFO] Starting cassandra-stress workload in background...
[INFO] Waiting 20s before injecting delay...
[INFO] Injecting delay to cassandra_b...
```



```
Results:
Op rate                   :    9,823 op/s  [WRITE: 9,823 op/s]
Partition rate            :    9,823 pk/s  [WRITE: 9,823 pk/s]
Row rate                  :    9,823 row/s [WRITE: 9,823 row/s]
Latency mean              :    5.0 ms [WRITE: 5.0 ms]
Latency median            :    0.8 ms [WRITE: 0.8 ms]
Latency 95th percentile   :   10.3 ms [WRITE: 10.3 ms]
Latency 99th percentile   :   10.4 ms [WRITE: 10.4 ms]
Latency 99.9th percentile :   17.8 ms [WRITE: 17.8 ms]
Latency max               :  195.6 ms [WRITE: 195.6 ms]
Total partitions          :    892,303 [WRITE: 892,303]
Total errors              :          0 [WRITE: 0]
Total GC count            : 0
Total GC memory           : 0.000 KiB
Total GC time             :    0.0 seconds
Avg GC time               :    NaN ms
StdDev GC time            :    0.0 ms
Total operation time      : 00:01:30

END
```

Each run used:

./run_netdelay.sh "cassandra_b cassandra_c" "<delay>" 30 delay<delay>ms_bc



```
./run_netdelay.sh "cassandra_b cassandra_c" "1ms" 30 delay1ms_bc
./run_netdelay.sh "cassandra_b cassandra_c" "50ms" 30 delay50ms_bc
./run_netdelay.sh "cassandra_b cassandra_c" "75ms" 30 delay75ms_bc
./run_netdelay.sh "cassandra_b cassandra_c" "100ms" 30 delay100ms_bc
```

Each script:

- Injects delay via `tc qdisc add ... delay Xms`
- Waits for stabilization
- Executes `cassandra-stress`
- Removes the delay via `tc qdisc del`

# Output Collection and Visualization

We parsed stress-output.log files to extract:

- Mean latency per second
- Ops/sec (throughput)

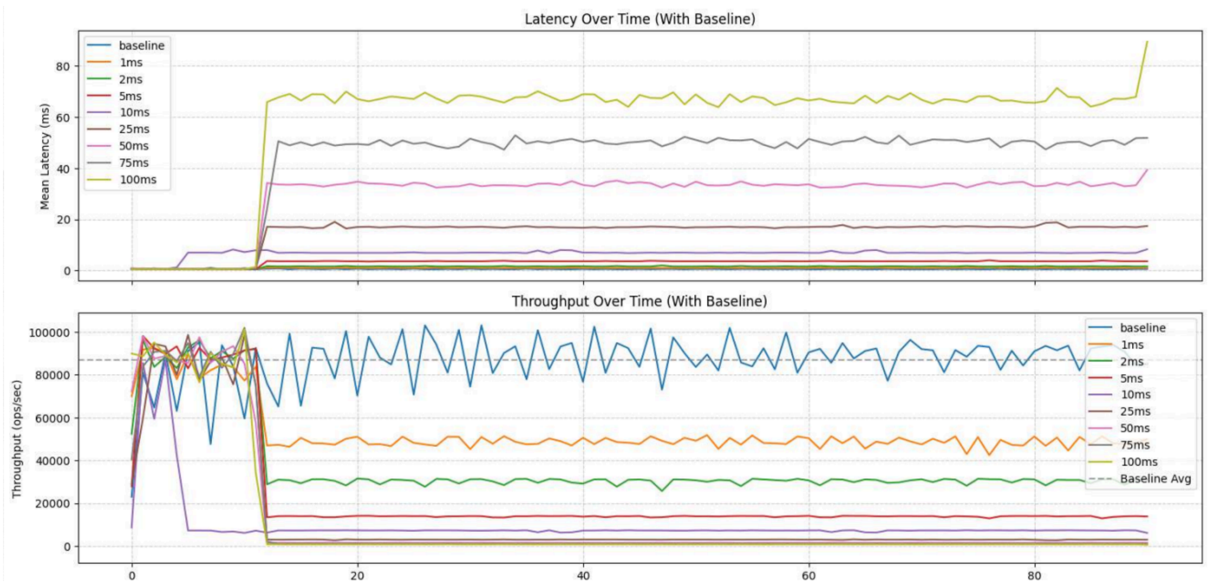
Then we visualized them using Python matplotlib, producing two subplots:

1. Latency over Time

2. Throughput over Time

Observations:

- At 1ms: ~50K ops/sec, ~1ms latency
- At 100ms: ~700 ops/sec, ~100ms latency
- 50ms and 75ms produce expected intermediate degradation

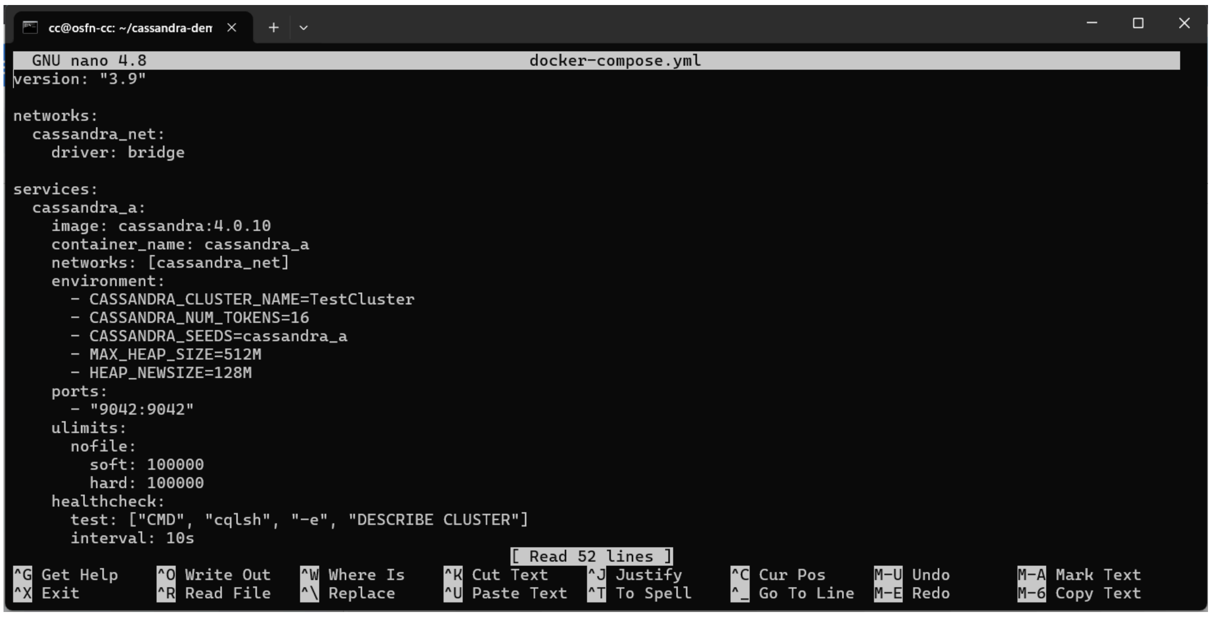


From the results, we can conclude that the experiment was unsuccessful due to several technical and experimental issues:

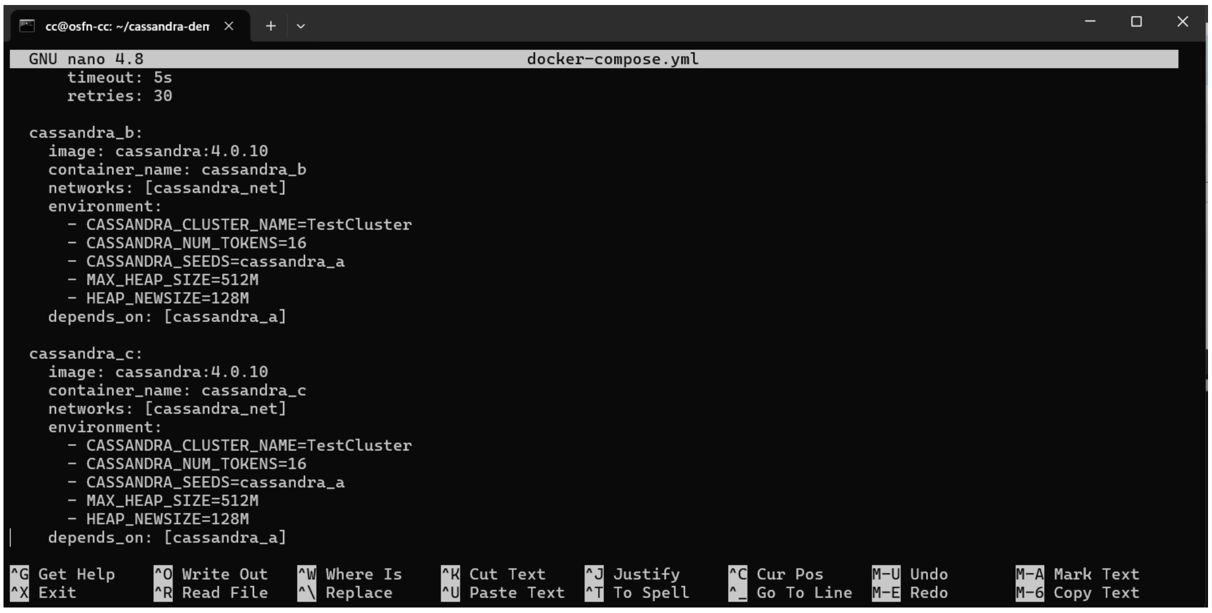
- Lack of observable performance impact: Across all delay configurations—from microseconds to seconds, both throughput and latency remained almost unchanged. This suggests that the injected faults did not effectively influence the system's behavior.
- Unexpected latency behavior: Higher delay values are expected to increase latency. However, the latency plots remained flat or even decreased, which contradicts expected behavior under network-level faults.
- Insufficient workload pressure: The `cassandra-stress` workload may have been too light, failing to sufficiently stress the system. As a result, even if delay was correctly injected, the system's performance remained unaffected.

Network Delay Injection using Blockade

Docker-compose.yml



```
  GNU nano 4.8                                    docker-compose.yml
version: "3.9"

networks:
  cassandra_net:
    driver: bridge

services:
  cassandra_a:
    image: cassandra:4.0.10
    container_name: cassandra_a
    networks: [cassandra_net]
    environment:
      - CASSANDRA_CLUSTER_NAME=TestCluster
      - CASSANDRA_NUM_TOKENS=16
      - CASSANDRA_SEEDS=cassandra_a
      - MAX_HEAP_SIZE=512M
      - HEAP_NEWSIZE=128M
    ports:
      - "9042:9042"
    ulimits:
      nofile:
        soft: 100000
        hard: 100000
    healthcheck:
      test: ["CMD", "cqlsh", "-e", "DESCRIBE CLUSTER"]
      interval: 10s
                                             [ Read 52 lines ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

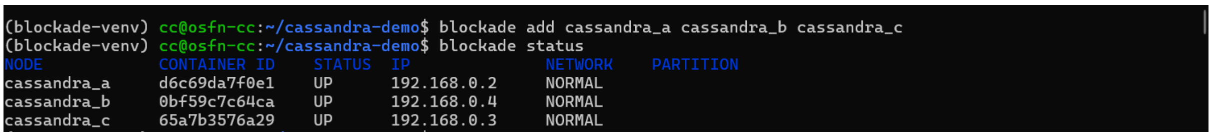


```
  GNU nano 4.8                                    docker-compose.yml
      timeout: 5s
      retries: 30

  cassandra_b:
    image: cassandra:4.0.10
    container_name: cassandra_b
    networks: [cassandra_net]
    environment:
      - CASSANDRA_CLUSTER_NAME=TestCluster
      - CASSANDRA_NUM_TOKENS=16
      - CASSANDRA_SEEDS=cassandra_a
      - MAX_HEAP_SIZE=512M
      - HEAP_NEWSIZE=128M
    depends_on: [cassandra_a]

  cassandra_c:
    image: cassandra:4.0.10
    container_name: cassandra_c
    networks: [cassandra_net]
    environment:
      - CASSANDRA_CLUSTER_NAME=TestCluster
      - CASSANDRA_NUM_TOKENS=16
      - CASSANDRA_SEEDS=cassandra_a
      - MAX_HEAP_SIZE=512M
      - HEAP_NEWSIZE=128M
    depends_on: [cassandra_a]

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

Blockade.yaml baseline

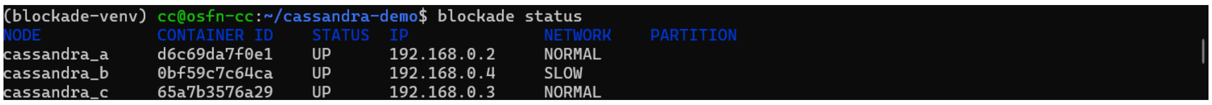
```
containers: {}   # we're going to "add" existing containers, so this can be empty
network:
  slow: 2ms                  # try 1–3ms to hit the "danger zone"
  flaky: 10%                 # handy for the etcd test later
  # driver: udn              # not needed when using `add`
```

```
blockade destroy || true
blockade add cassandra_a cassandra_b cassandra_c
blockade status
```



```
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ blockade add cassandra_a cassandra_b cassandra_c
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ blockade status
NODE            CONTAINER ID    STATUS  IP              NETWORK    PARTITION
cassandra_a     d6c69da7f0e1    UP      192.168.0.2     NORMAL
cassandra_b     0bf59c7c64ca    UP      192.168.0.4     NORMAL
cassandra_c     65a7b3576a29    UP      192.168.0.3     NORMAL
```

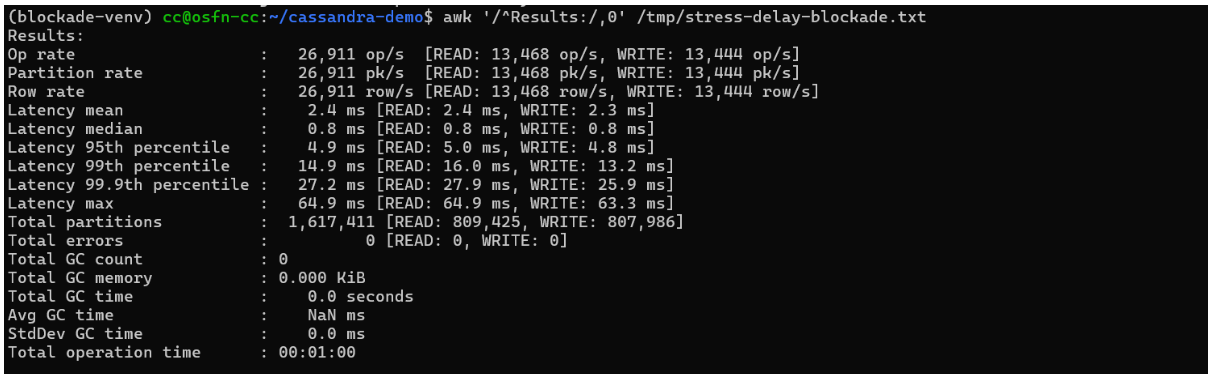
blockade slow cassandra_b
blockade status



```
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ blockade status
NODE            CONTAINER ID    STATUS  IP              NETWORK    PARTITION
cassandra_a     d6c69da7f0e1    UP      192.168.0.2     NORMAL
cassandra_b     0bf59c7c64ca    UP      192.168.0.4     SLOW
cassandra_c     65a7b3576a29    UP      192.168.0.3     NORMAL
```

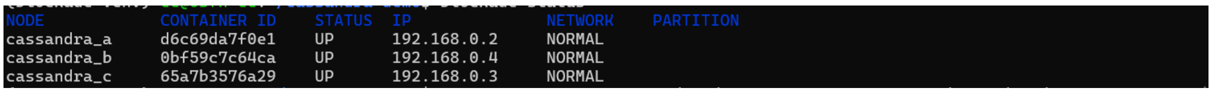
docker exec cassandra_a /tmp/apache-cassandra-4.0.10/tools/bin/cassandra-stress \
> mixed 'ratio(write=1,read=1)' duration=60s cl=QUORUM \
> -pop seq=1..200000 \
> -node cassandra_a,cassandra_b,cassandra_c \
> -rate threads=64 -mode native cql3 > /tmp/stress-delay-blockade.txt 2>&1

awk '/^Results:/,0' /tmp/stress-delay-blockade.txt



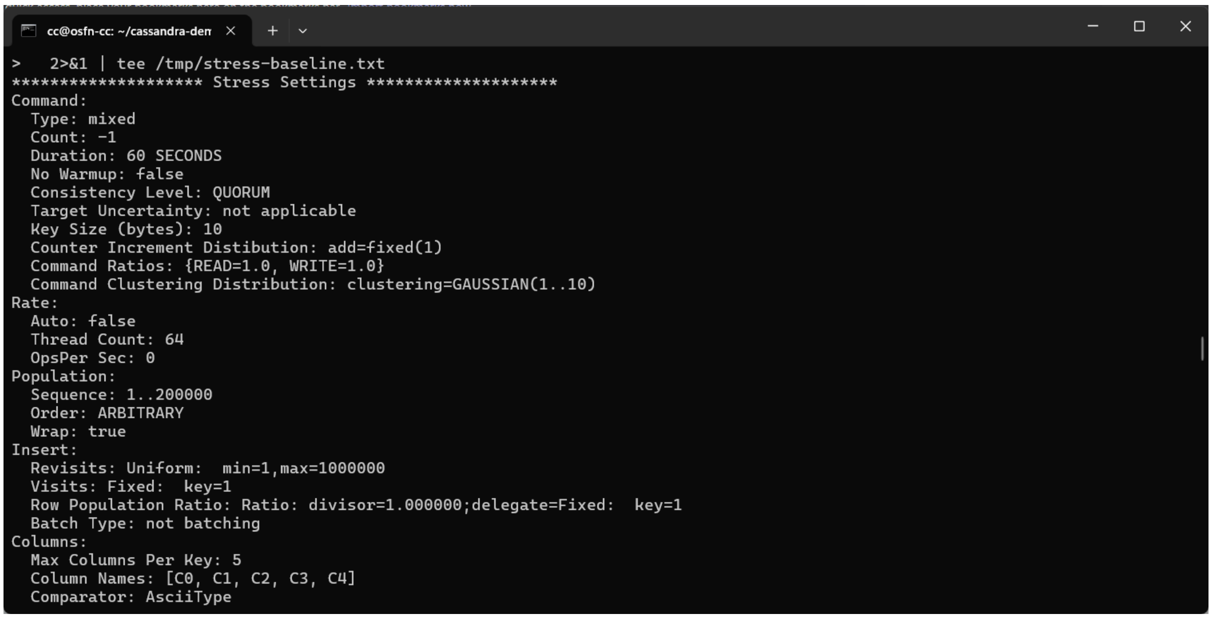
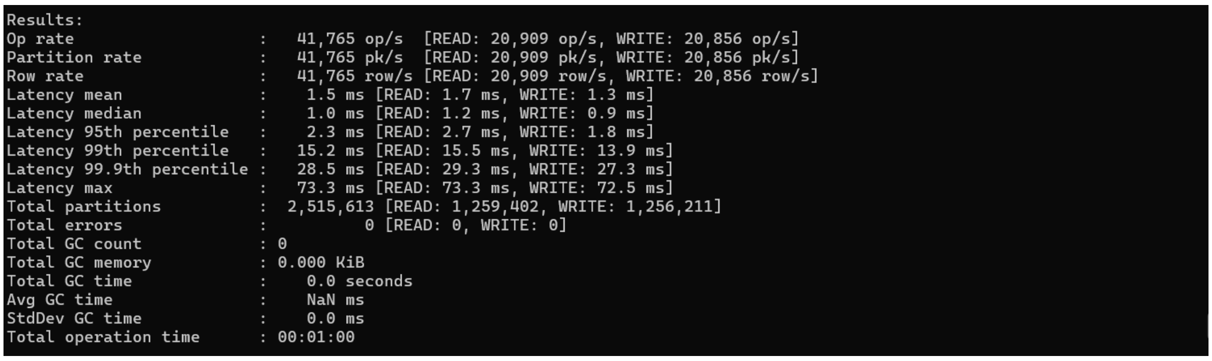
```
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ awk '/^Results:/,0' /tmp/stress-delay-blockade.txt
Results:
Op rate                   :   26,911 op/s  [READ: 13,468 op/s, WRITE: 13,444 op/s]
Partition rate            :   26,911 pk/s  [READ: 13,468 pk/s, WRITE: 13,444 pk/s]
Row rate                  :   26,911 row/s [READ: 13,468 row/s, WRITE: 13,444 row/s]
Latency mean              :    2.4 ms [READ: 2.4 ms, WRITE: 2.3 ms]
Latency median            :    0.8 ms [READ: 0.8 ms, WRITE: 0.8 ms]
Latency 95th percentile   :    4.9 ms [READ: 5.0 ms, WRITE: 4.8 ms]
Latency 99th percentile   :   14.9 ms [READ: 16.0 ms, WRITE: 13.2 ms]
Latency 99.9th percentile :   27.2 ms [READ: 27.9 ms, WRITE: 25.9 ms]
Latency max               :   64.9 ms [READ: 64.9 ms, WRITE: 63.3 ms]
Total partitions          :  1,617,411 [READ: 809,425, WRITE: 807,986]
Total errors              :          0 [READ: 0, WRITE: 0]
Total GC count            : 0
Total GC memory           : 0.000 KiB
Total GC time             :    0.0 seconds
Avg GC time               :    NaN ms
StdDev GC time            :    0.0 ms
Total operation time      : 00:01:00
```

Running Baseline
blockade fast --all
blockade status



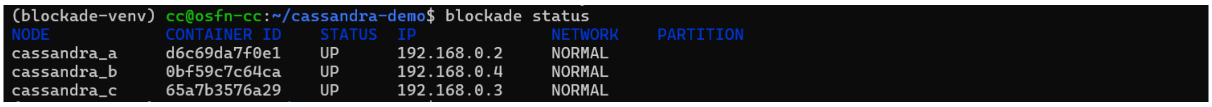
```
NODE            CONTAINER ID    STATUS  IP              NETWORK    PARTITION
cassandra_a     d6c69da7f0e1    UP      192.168.0.2     NORMAL
cassandra_b     0bf59c7c64ca    UP      192.168.0.4     NORMAL
cassandra_c     65a7b3576a29    UP      192.168.0.3     NORMAL
```

(blockade-venv) cc@osfn-cc:~/cassandra-demo$ docker exec cassandra_a /tmp/apache-cassandra-4.0.10/tools/bin/cassandra-stress \
> mixed 'ratio(write=1,read=1)' duration=60s cl=QUORUM \
> -pop seq=1..200000 \
> -node cassandra_a,cassandra_b,cassandra_c \
> -rate threads=64 -mode native cql3 \
> 2>&1 | tee /tmp/stress-baseline.txt


```
>   2>&1 | tee /tmp/stress-baseline.txt
******************** Stress Settings ********************
Command:
  Type: mixed
  Count: -1
  Duration: 60 SECONDS
  No Warmup: false
  Consistency Level: QUORUM
  Target Uncertainty: not applicable
  Key Size (bytes): 10
  Counter Increment Distibution: add=fixed(1)
  Command Ratios: {READ=1.0, WRITE=1.0}
  Command Clustering Distribution: clustering=GAUSSIAN(1..10)
Rate:
  Auto: false
  Thread Count: 64
  OpsPer Sec: 0
Population:
  Sequence: 1..200000
  Order: ARBITRARY
  Wrap: true
Insert:
  Revisits: Uniform:  min=1,max=1000000
  Visits: Fixed:  key=1
  Row Population Ratio: Ratio: divisor=1.000000;delegate=Fixed:  key=1
  Batch Type: not batching
Columns:
  Max Columns Per Key: 5
  Column Names: [C0, C1, C2, C3, C4]
  Comparator: AsciiType
```

```
Results:
Op rate                   :   41,765 op/s  [READ: 20,909 op/s, WRITE: 20,856 op/s]
Partition rate            :   41,765 pk/s  [READ: 20,909 pk/s, WRITE: 20,856 pk/s]
Row rate                  :   41,765 row/s [READ: 20,909 row/s, WRITE: 20,856 row/s]
Latency mean              :    1.5 ms [READ: 1.7 ms, WRITE: 1.3 ms]
Latency median            :    1.0 ms [READ: 1.2 ms, WRITE: 0.9 ms]
Latency 95th percentile   :    2.3 ms [READ: 2.7 ms, WRITE: 1.8 ms]
Latency 99th percentile   :   15.2 ms [READ: 15.5 ms, WRITE: 13.9 ms]
Latency 99.9th percentile :   28.5 ms [READ: 29.3 ms, WRITE: 27.3 ms]
Latency max               :   73.3 ms [READ: 73.3 ms, WRITE: 72.5 ms]
Total partitions          :  2,515,613 [READ: 1,259,402, WRITE: 1,256,211]
Total errors              :          0 [READ: 0, WRITE: 0]
Total GC count            : 0
Total GC memory           : 0.000 KiB
Total GC time             :    0.0 seconds
Avg GC time               :    NaN ms
StdDev GC time            :    0.0 ms
Total operation time      : 00:01:00
```

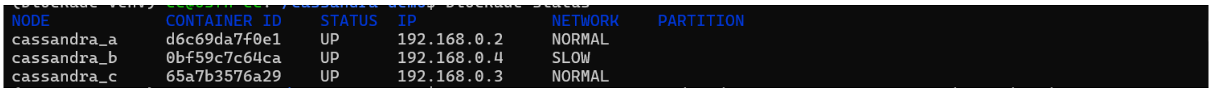
Running slow delay

blockade status

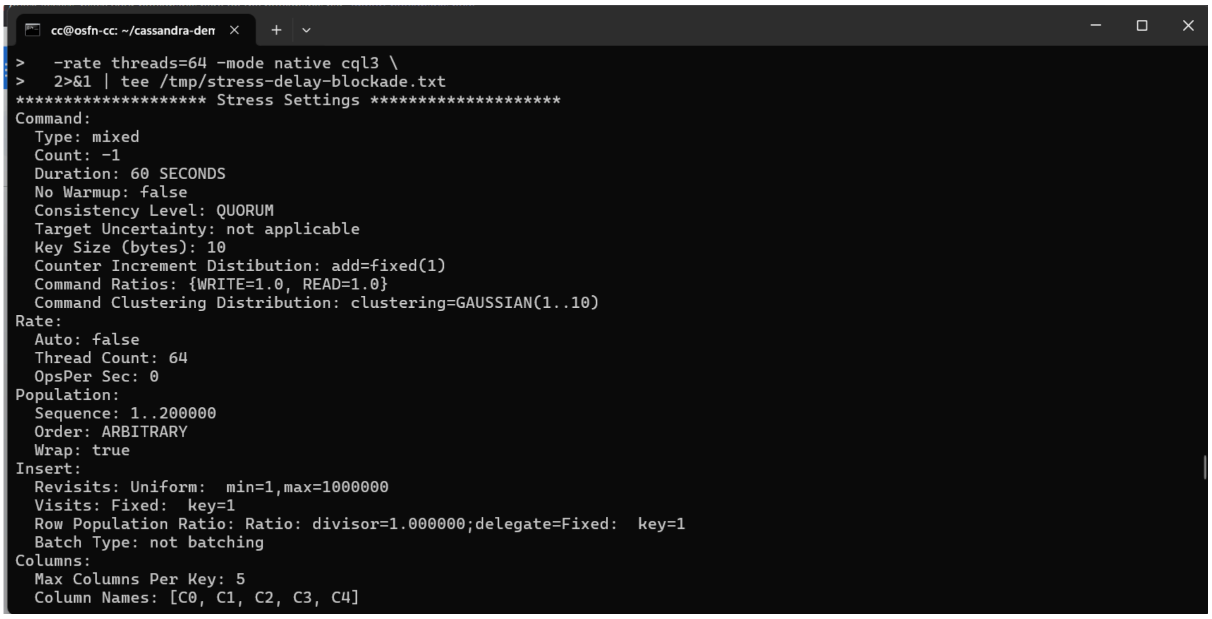

```
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ blockade status
NODE            CONTAINER ID    STATUS  IP              NETWORK    PARTITION
cassandra_a     d6c69da7f0e1    UP      192.168.0.2     NORMAL
cassandra_b     0bf59c7c64ca    UP      192.168.0.4     NORMAL
cassandra_c     65a7b3576a29    UP      192.168.0.3     NORMAL
```

blockade slow cassandra_b

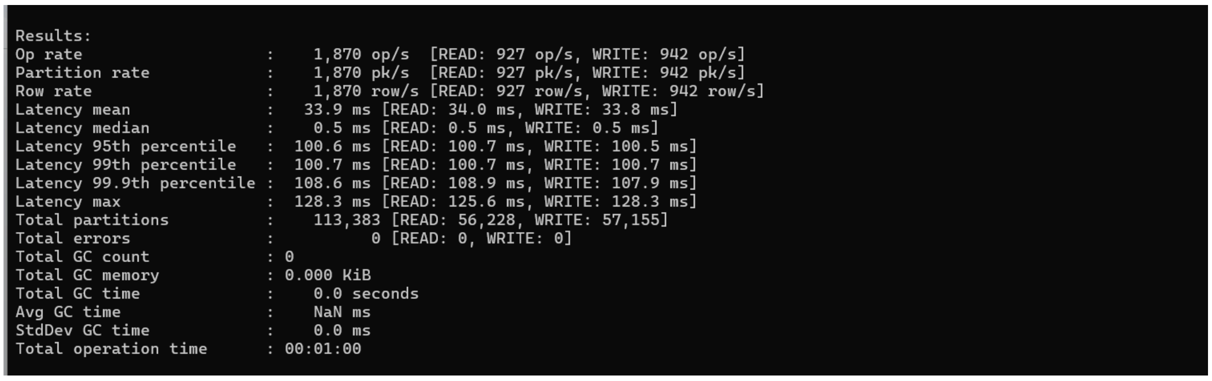
blockade status


```
NODE            CONTAINER ID    STATUS  IP              NETWORK    PARTITION
cassandra_a     d6c69da7f0e1    UP      192.168.0.2     NORMAL
cassandra_b     0bf59c7c64ca    UP      192.168.0.4     SLOW
cassandra_c     65a7b3576a29    UP      192.168.0.3     NORMAL
```

```
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ docker exec cassandra_a
/tmp/apache-cassandra-4.0.10/tools/bin/cassandra-stress \
>   mixed 'ratio(write=1,read=1)' duration=60s cl=QUORUM \
>   -pop seq=1..200000 \
>   -node cassandra_a,cassandra_b,cassandra_c \
>   -rate threads=64 -mode native cql3 \
>   2>&1 | tee /tmp/stress-delay-blockade.txt
```

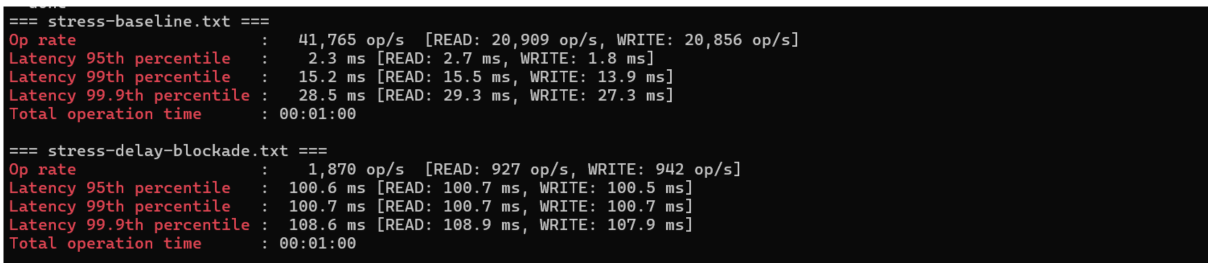


```
>   -rate threads=64 -mode native cql3 \
>   2>&1 | tee /tmp/stress-delay-blockade.txt
******************** Stress Settings ********************
Command:
  Type: mixed
  Count: -1
  Duration: 60 SECONDS
  No Warmup: false
  Consistency Level: QUORUM
  Target Uncertainty: not applicable
  Key Size (bytes): 10
  Counter Increment Distibution: add=fixed(1)
  Command Ratios: {WRITE=1.0, READ=1.0}
  Command Clustering Distribution: clustering=GAUSSIAN(1..10)
Rate:
  Auto: false
  Thread Count: 64
  OpsPer Sec: 0
Population:
  Sequence: 1..200000
  Order: ARBITRARY
  Wrap: true
Insert:
  Revisits: Uniform:  min=1,max=1000000
  Visits: Fixed:  key=1
  Row Population Ratio: Ratio: divisor=1.000000;delegate=Fixed:  key=1
  Batch Type: not batching
Columns:
  Max Columns Per Key: 5
  Column Names: [C0, C1, C2, C3, C4]
```



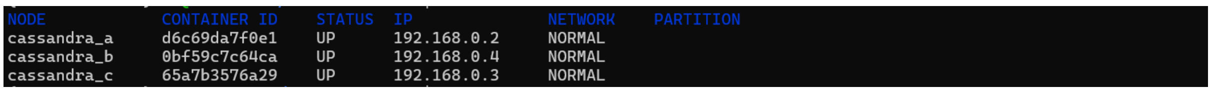
```
Results:
Op rate                   :    1,870 op/s  [READ: 927 op/s, WRITE: 942 op/s]
Partition rate            :    1,870 pk/s  [READ: 927 pk/s, WRITE: 942 pk/s]
Row rate                  :    1,870 row/s [READ: 927 row/s, WRITE: 942 row/s]
Latency mean              :   33.9 ms [READ: 34.0 ms, WRITE: 33.8 ms]
Latency median            :    0.5 ms [READ: 0.5 ms, WRITE: 0.5 ms]
Latency 95th percentile   :  100.6 ms [READ: 100.7 ms, WRITE: 100.5 ms]
Latency 99th percentile   :  100.7 ms [READ: 100.7 ms, WRITE: 100.7 ms]
Latency 99.9th percentile :  108.6 ms [READ: 108.9 ms, WRITE: 107.9 ms]
Latency max               :  128.3 ms [READ: 125.6 ms, WRITE: 128.3 ms]
Total partitions          :    113,383 [READ: 56,228, WRITE: 57,155]
Total errors              :          0 [READ: 0, WRITE: 0]
Total GC count            : 0
Total GC memory           : 0.000 KiB
Total GC time             :    0.0 seconds
Avg GC time               :    NaN ms
StdDev GC time            :    0.0 ms
Total operation time      : 00:01:00
```

```
for f in /tmp/stress-baseline.txt /tmp/stress-delay-blockade.txt; do
>   echo "=== $(basename "$f") ==="
>   grep -E 'Op rate|Latency 95th percentile|Latency 99th percentile|Latency 99.9th percentile|Total operation time' "$f"
>   echo
> done
```



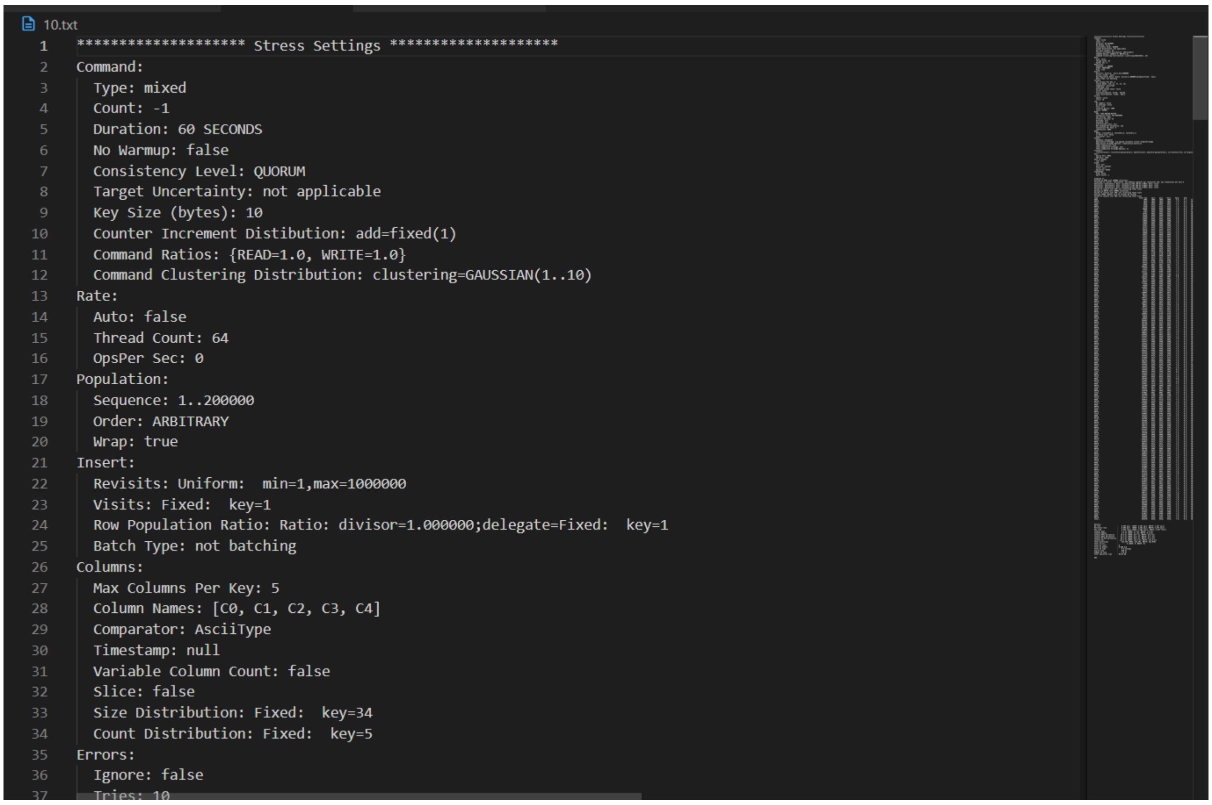
```
=== stress-baseline.txt ===
Op rate                   :   41,765 op/s  [READ: 20,909 op/s, WRITE: 20,856 op/s]
Latency 95th percentile   :    2.3 ms [READ: 2.7 ms, WRITE: 1.8 ms]
Latency 99th percentile   :   15.2 ms [READ: 15.5 ms, WRITE: 13.9 ms]
Latency 99.9th percentile :   28.5 ms [READ: 29.3 ms, WRITE: 27.3 ms]
Total operation time      : 00:01:00

=== stress-delay-blockade.txt ===
Op rate                   :    1,870 op/s  [READ: 927 op/s, WRITE: 942 op/s]
Latency 95th percentile   :  100.6 ms [READ: 100.7 ms, WRITE: 100.5 ms]
Latency 99th percentile   :  100.7 ms [READ: 100.7 ms, WRITE: 100.7 ms]
Latency 99.9th percentile :  108.6 ms [READ: 108.9 ms, WRITE: 107.9 ms]
Total operation time      : 00:01:00
```

```
blockade fast cassandra_b
blockade status
```

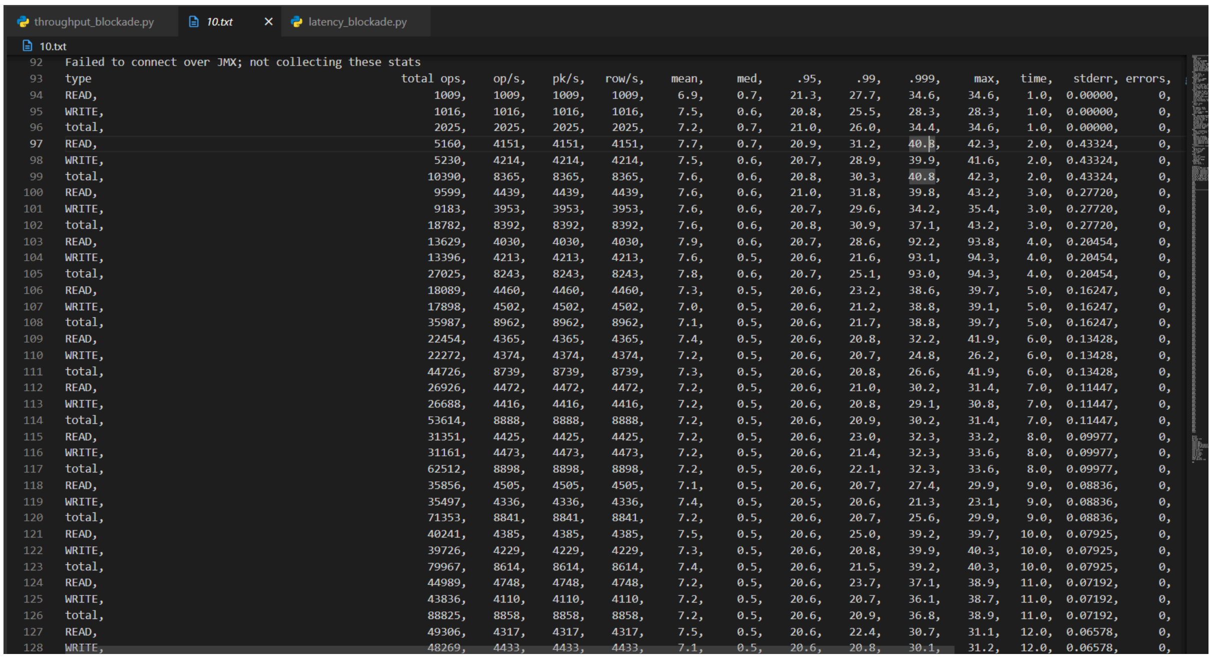


```
NODE          CONTAINER ID    STATUS  IP             NETWORK    PARTITION
cassandra_a   d6c69da7f0e1    UP      192.168.0.2    NORMAL
cassandra_b   0bf59c7c64ca    UP      192.168.0.4    NORMAL
cassandra_c   65a7b3576a29    UP      192.168.0.3    NORMAL
```

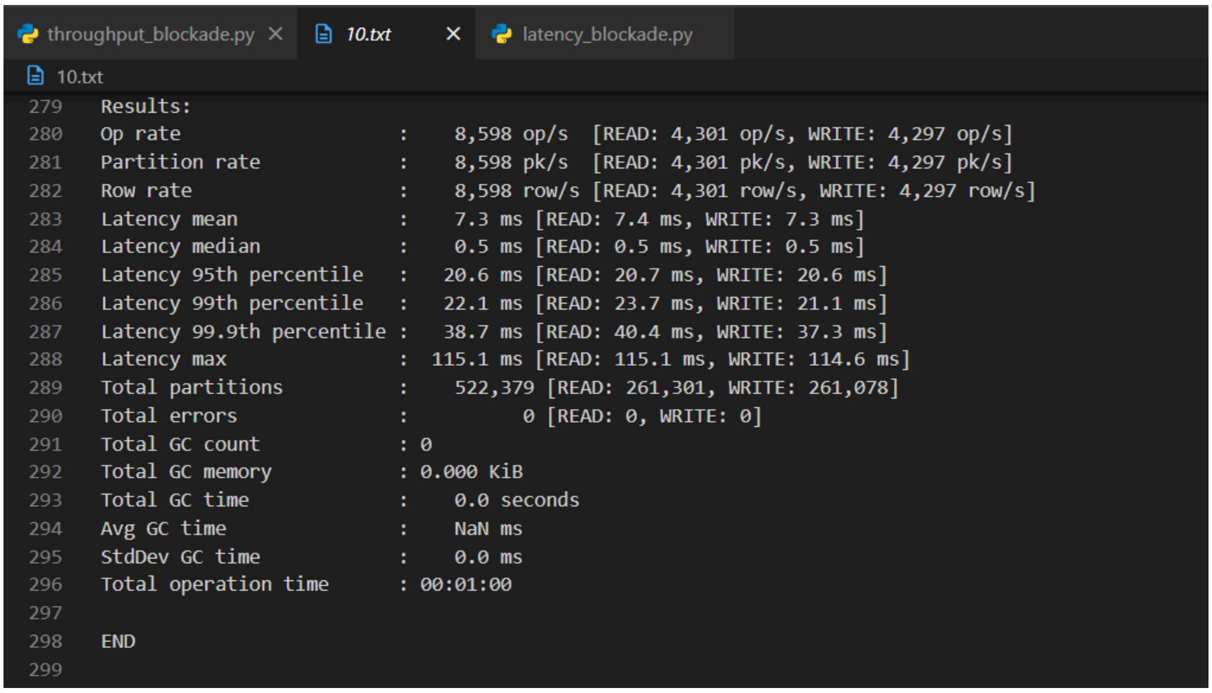
File results for the delay



```
******************** Stress Settings ********************
Command:
  Type: mixed
  Count: -1
  Duration: 60 SECONDS
  No Warmup: false
  Consistency Level: QUORUM
  Target Uncertainty: not applicable
  Key Size (bytes): 10
  Counter Increment Distibution: add=fixed(1)
  Command Ratios: {READ=1.0, WRITE=1.0}
  Command Clustering Distribution: clustering=GAUSSIAN(1..10)
Rate:
  Auto: false
  Thread Count: 64
  OpsPer Sec: 0
Population:
  Sequence: 1..200000
  Order: ARBITRARY
  Wrap: true
Insert:
  Revisits: Uniform:  min=1,max=1000000
  Visits: Fixed:  key=1
  Row Population Ratio: Ratio: divisor=1.000000;delegate=Fixed:  key=1
  Batch Type: not batching
Columns:
  Max Columns Per Key: 5
  Column Names: [C0, C1, C2, C3, C4]
  Comparator: AsciiType
  Timestamp: null
  Variable Column Count: false
  Slice: false
  Size Distribution: Fixed:  key=34
  Count Distribution: Fixed:  key=5
Errors:
  Ignore: false
  Tries: 10
```

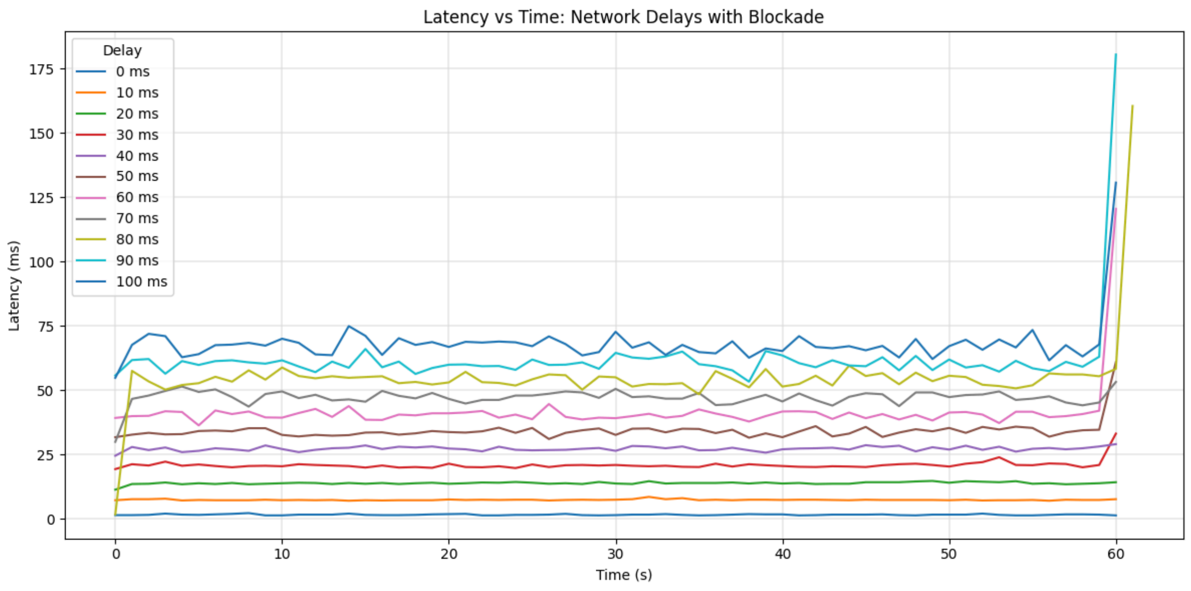


```
Failed to connect over JMX; not collecting these stats
type                                      total ops,   op/s,   pk/s,  row/s,   mean,    med,    .95,    .99,   .999,    max,   time,   stderr, errors,
READ,                                          1009,   1009,   1009,   1009,    6.9,    0.7,   21.3,   27.7,   34.6,   34.6,    1.0,  0.00000,      0,
WRITE,                                         1016,   1016,   1016,   1016,    7.5,    0.6,   20.8,   25.5,   28.3,   28.3,    1.0,  0.00000,      0,
total,                                         2025,   2025,   2025,   2025,    7.2,    0.7,   21.0,   26.0,   34.4,   34.6,    1.0,  0.00000,      0,
READ,                                          5160,   4151,   4151,   4151,    7.7,    0.7,   20.9,   31.2,   40.8,   42.3,    2.0,  0.43324,      0,
WRITE,                                         5230,   4214,   4214,   4214,    7.5,    0.6,   20.7,   28.9,   39.9,   41.6,    2.0,  0.43324,      0,
total,                                        10390,   8365,   8365,   8365,    7.6,    0.6,   20.8,   30.3,   40.8,   42.3,    2.0,  0.43324,      0,
READ,                                          9599,   4439,   4439,   4439,    7.6,    0.6,   21.0,   31.8,   39.8,   43.2,    3.0,  0.27720,      0,
WRITE,                                         9183,   3953,   3953,   3953,    7.6,    0.6,   20.7,   29.6,   34.2,   35.4,    3.0,  0.27720,      0,
total,                                        18782,   8392,   8392,   8392,    7.6,    0.6,   20.8,   30.9,   37.1,   43.2,    3.0,  0.27720,      0,
READ,                                         13629,   4030,   4030,   4030,    7.9,    0.6,   20.7,   28.6,   92.2,   93.8,    4.0,  0.20454,      0,
WRITE,                                        13396,   4213,   4213,   4213,    7.6,    0.5,   20.6,   21.6,   93.1,   94.3,    4.0,  0.20454,      0,
total,                                        27025,   8243,   8243,   8243,    7.8,    0.6,   20.7,   25.1,   93.0,   94.3,    4.0,  0.20454,      0,
READ,                                         18089,   4460,   4460,   4460,    7.3,    0.5,   20.6,   23.2,   38.6,   39.7,    5.0,  0.16247,      0,
WRITE,                                        17898,   4502,   4502,   4502,    7.0,    0.5,   20.6,   21.2,   38.8,   39.1,    5.0,  0.16247,      0,
total,                                        35987,   8962,   8962,   8962,    7.1,    0.5,   20.6,   21.7,   38.8,   39.7,    5.0,  0.16247,      0,
READ,                                         22454,   4365,   4365,   4365,    7.4,    0.5,   20.6,   20.8,   32.2,   41.9,    6.0,  0.13428,      0,
WRITE,                                        22272,   4374,   4374,   4374,    7.2,    0.5,   20.6,   20.7,   24.8,   26.2,    6.0,  0.13428,      0,
total,                                        44726,   8739,   8739,   8739,    7.3,    0.5,   20.6,   20.8,   26.6,   41.9,    6.0,  0.13428,      0,
READ,                                         26926,   4472,   4472,   4472,    7.2,    0.5,   20.6,   21.0,   30.2,   31.4,    7.0,  0.11447,      0,
WRITE,                                        26688,   4416,   4416,   4416,    7.2,    0.5,   20.6,   20.8,   29.1,   30.8,    7.0,  0.11447,      0,
total,                                        53614,   8888,   8888,   8888,    7.2,    0.5,   20.6,   20.9,   30.2,   31.4,    7.0,  0.11447,      0,
READ,                                         31351,   4425,   4425,   4425,    7.2,    0.5,   20.6,   23.0,   32.3,   33.2,    8.0,  0.09977,      0,
WRITE,                                        31161,   4473,   4473,   4473,    7.2,    0.5,   20.6,   21.4,   32.3,   33.6,    8.0,  0.09977,      0,
total,                                        62512,   8898,   8898,   8898,    7.2,    0.5,   20.6,   22.1,   32.3,   33.6,    8.0,  0.09977,      0,
READ,                                         35856,   4505,   4505,   4505,    7.1,    0.5,   20.6,   20.7,   27.4,   29.9,    9.0,  0.08836,      0,
WRITE,                                        35497,   4336,   4336,   4336,    7.4,    0.5,   20.5,   20.6,   21.3,   23.1,    9.0,  0.08836,      0,
total,                                        71353,   8841,   8841,   8841,    7.2,    0.5,   20.6,   20.7,   25.6,   29.9,    9.0,  0.08836,      0,
READ,                                         40241,   4385,   4385,   4385,    7.5,    0.5,   20.6,   25.0,   39.2,   39.7,   10.0,  0.07925,      0,
WRITE,                                        39726,   4229,   4229,   4229,    7.3,    0.5,   20.6,   20.8,   39.9,   40.3,   10.0,  0.07925,      0,
total,                                        79967,   8614,   8614,   8614,    7.4,    0.5,   20.6,   21.5,   39.2,   40.3,   10.0,  0.07925,      0,
READ,                                         44989,   4748,   4748,   4748,    7.2,    0.5,   20.6,   23.7,   37.1,   38.9,   11.0,  0.07192,      0,
WRITE,                                        43836,   4110,   4110,   4110,    7.2,    0.5,   20.6,   20.7,   36.1,   38.7,   11.0,  0.07192,      0,
total,                                        88825,   8858,   8858,   8858,    7.2,    0.5,   20.6,   20.9,   36.8,   38.9,   11.0,  0.07192,      0,
READ,                                         49306,   4317,   4317,   4317,    7.5,    0.5,   20.6,   22.4,   30.7,   31.1,   12.0,  0.06578,      0,
WRITE,                                        48269,   4433,   4433,   4433,    7.1,    0.5,   20.6,   20.8,   30.1,   31.2,   12.0,  0.06578,      0,
```

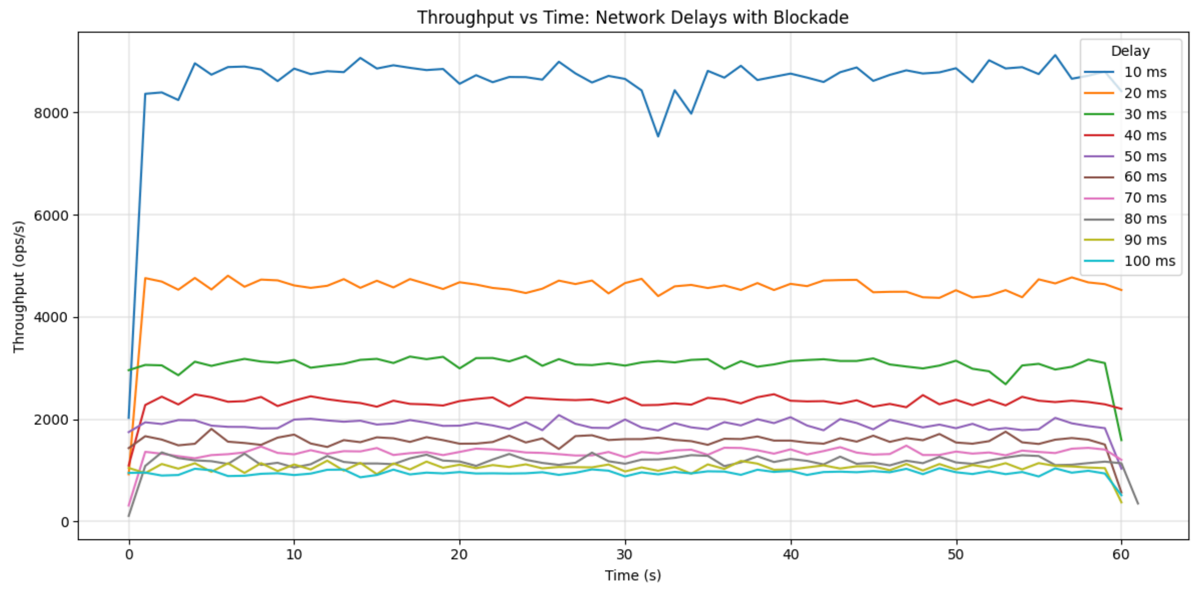


```
Results:
Op rate                   :    8,598 op/s  [READ: 4,301 op/s, WRITE: 4,297 op/s]
Partition rate            :    8,598 pk/s  [READ: 4,301 pk/s, WRITE: 4,297 pk/s]
Row rate                  :    8,598 row/s [READ: 4,301 row/s, WRITE: 4,297 row/s]
Latency mean              :    7.3 ms [READ: 7.4 ms, WRITE: 7.3 ms]
Latency median            :    0.5 ms [READ: 0.5 ms, WRITE: 0.5 ms]
Latency 95th percentile   :   20.6 ms [READ: 20.7 ms, WRITE: 20.6 ms]
Latency 99th percentile   :   22.1 ms [READ: 23.7 ms, WRITE: 21.1 ms]
Latency 99.9th percentile :   38.7 ms [READ: 40.4 ms, WRITE: 37.3 ms]
Latency max               :  115.1 ms [READ: 115.1 ms, WRITE: 114.6 ms]
Total partitions          :    522,379 [READ: 261,301, WRITE: 261,078]
Total errors              :          0 [READ: 0, WRITE: 0]
Total GC count            : 0
Total GC memory           : 0.000 KiB
Total GC time             :    0.0 seconds
Avg GC time               :    NaN ms
StdDev GC time            :    0.0 ms
Total operation time      : 00:01:00

END
```

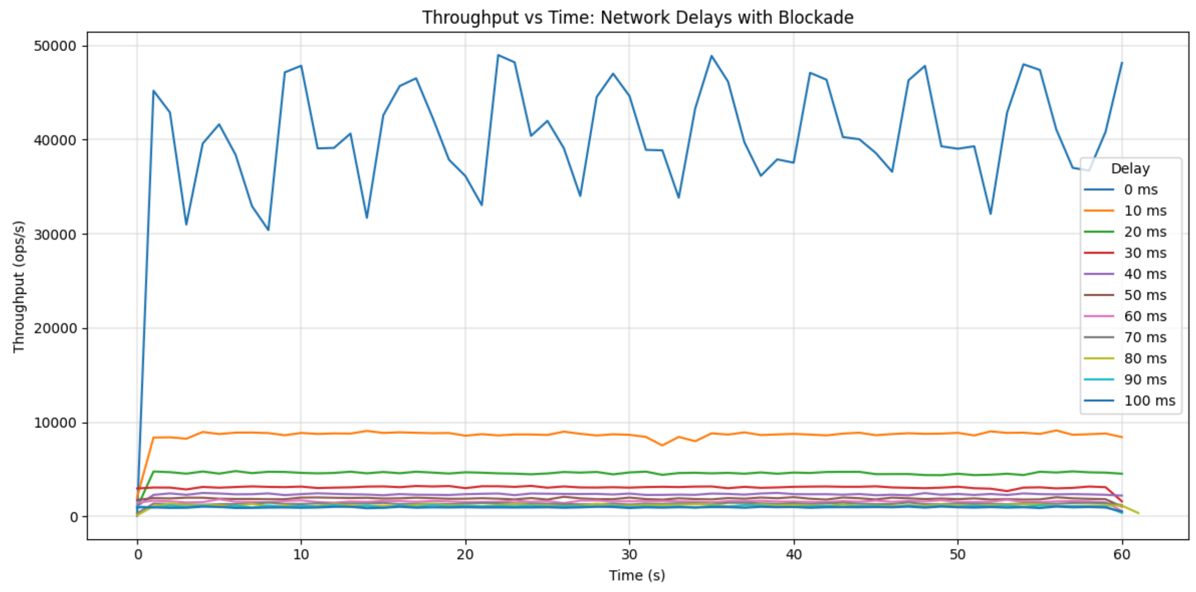
Latency vs Time



Throughput vs Time



With baseline


Throughput vs Time: Network Delays with Blockade
Throughput (ops/s)
Time (s)
Delay
0 ms
10 ms
20 ms
30 ms
40 ms
50 ms
60 ms
70 ms
80 ms
90 ms
100 ms