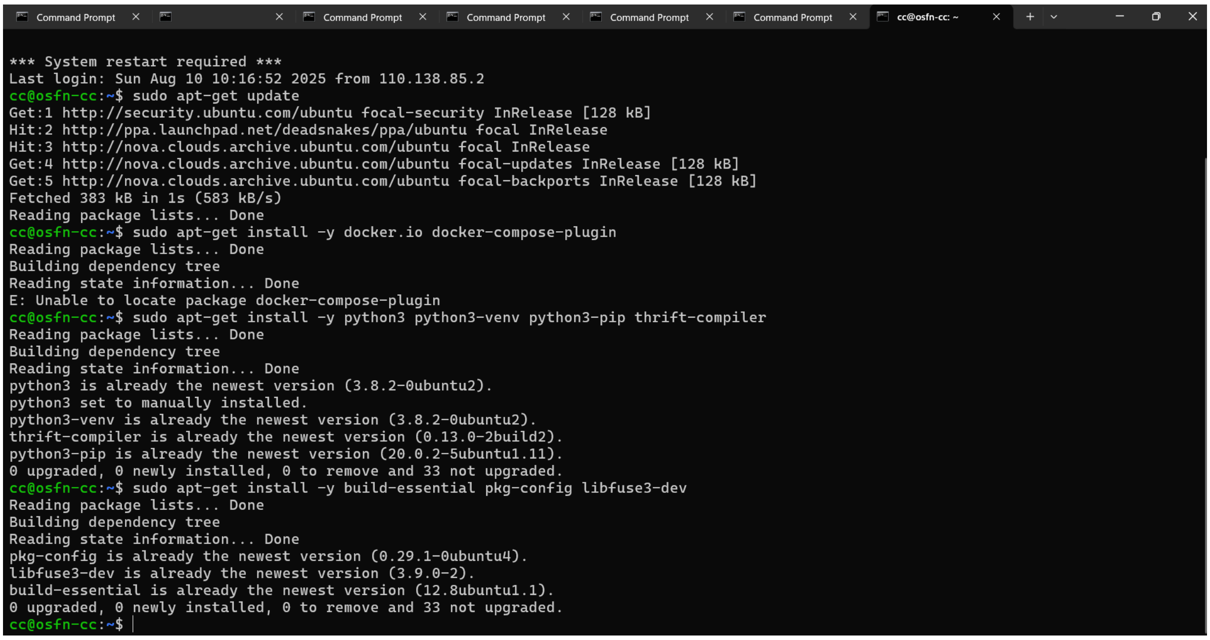
Reproducing Filesystem Delay Experiments: One-Size-Fits-None (NSDI'25)

System Requirements Prerequisites (Ubuntu 20.04/22.04 tested)

```
# Docker + Compose v2
sudo apt-get update
sudo apt-get install -y docker.io docker-compose-plugin

# Python 3 + thrift compiler (Charybdefs client uses Thrift)
sudo apt-get install -y python3 python3-venv python3-pip
thrift-compiler

# FUSE3 dev (to build Charybdefs) - if built locally
sudo apt-get install -y build-essential pkg-config libfuse3-dev
```



1) Charybdefs: build, mount, and management service

The paper uses Charybdefs to inject filesystem slowness via a (Thrift) management interface.

If you already have Charybdefs built and mounted, skip to Step 2 and just keep the paths the same.

```
# Build (any clean location is fine)
cd ~
```

```
git clone https://github.com/scylladb/charybdefs.git
cd charybdefs

# builds the FUSE FS and the mgmt server
make
```

```
cc@osfn-cc:~$
cc@osfn-cc:~$ ls
CHI-210850-openrc.sh  blockade--venv       blockade-venv   charybdefs  my_mounting_point  sweep_logs  traces
README                blockade-py36-venv   cassandra-demo  logs        openrc             trace-ACER
cc@osfn-cc:~$ cd charybdefs
cc@osfn-cc:~/charybdefs$ make
[ 20%] Built target faultcore
[ 33%] Built target charybde_lib
[ 40%] Built target thrift_gen
[ 73%] Built target server
[ 86%] Built target server_lib
[100%] Built target charybdefs
cc@osfn-cc:~/charybdefs$
```

```
# Create a mount point that will back the etcd data dirs
sudo mkdir -p /mnt/slowfs/{etcd0,etcd1,etcd2}

# Run Charybdefs (FS + mgmt on 0.0.0.0:9090).
# The typical pattern is one charybdefs instance serving a root
mount and exposing a Thrift RPC on 9090.
# Adjust the exact binary/flags if your tree differs (some builds
split FS and mgmt).
sudo ./build/charybdefs -o allow_other -m /mnt/slowfs &

# start mgmt (Thrift) — name/flags vary slightly by branch; if
unified binary, omit this
./build/charybdefs-mgmt --bind 0.0.0.0 --port 9090 &
```

```
cc@osfn-cc:~/charybdefs$ sudo mkdir -p /mnt/slowfs/{etcd0,etcd1,etcd2}
cc@osfn-cc:~/charybdefs$ sudo ./build/charybdefs -o allow_other -m /mnt/slowfs &
[1] 2489737
```

- We'll point etcd's data directories to `/mnt/slowfs/etcd{0,1,2}` so **every WAL/fsync** route goes through Charybdefs and can be delayed.
- The mgmt endpoint is Thrift on TCP :9090 (as in the paper).

2) Bring up a 3-node etcd cluster (Docker Compose)

 # Bring up the cluster

```
cd ~/cassandra-demo
docker compose -f docker-compose-etcd.yml up -d
```

```
cc@osfn-cc:~/cassandra-demo$ docker compose -f docker-compose-etcd.yml up -d
[+] Building 0.0s (0/0)
[+] Running 4/4
 ✔Network cassandra-demo_default  Created                                                              0.2s
 ✔Container etcd0                 Started                                                              0.4s
 ✔Container etcd1                 Started                                                              0.4s
 ✔Container etcd2                 Started                                                              0.3s
cc@osfn-cc:~/cassandra-demo$ |
```

Set a convenience variable for etcdctl

```
export
ENDPOINTS="http://etcd0:2379,http://etcd1:2379,http://etcd2:2379"
```

Health check

```
docker exec etcd1 /usr/local/bin/etcdctl --endpoints="$ENDPOINTS"
endpoint status -w table
```

```
cc@osfn-cc:~/cassandra-demo$ export ENDPOINTS="http://etcd0:2379,http://etcd1:2379,http://etcd2:2379"
cc@osfn-cc:~/cassandra-demo$ docker exec etcd1 /usr/local/bin/etcdctl --endpoints="$ENDPOINTS" endpoint status -w table
+-------------------+------------------+---------+-----------------+---------+--------+-----------------------+-------+-----------+------------+
-----------+------------+--------------------+--------+--------------------------+-------------------+
|     ENDPOINT      |        ID        | VERSION | STORAGE VERSION | DB SIZE | IN USE | PERCENTAGE NOT IN USE | QUOTA | IS LEADER | IS LEARNER |
 RAFT TERM | RAFT INDEX | RAFT APPLIED INDEX | ERRORS | DOWNGRADE TARGET VERSION | DOWNGRADE ENABLED |
+-------------------+------------------+---------+-----------------+---------+--------+-----------------------+-------+-----------+------------+
-----------+------------+--------------------+--------+--------------------------+-------------------+
| http://etcd0:2379 | cf1d15c5d194b5c9 |   3.6.2 |           3.6.0 |  637 MB | 628 MB |                    2% |   0 B |     false |      false |
        24 |     135578 |             135578 |        |                          |             false |
| http://etcd1:2379 | ade526d28b1f92f7 |   3.6.2 |           3.6.0 |  637 MB | 628 MB |                    2% |   0 B |      true |      false |
        24 |     135578 |             135578 |        |                          |             false |
| http://etcd2:2379 | d282ac2ce600c1ce |   3.6.2 |           3.6.0 |  637 MB | 628 MB |                    2% |   0 B |     false |      false |
        24 |     135578 |             135578 |        |                          |             false |
+-------------------+------------------+---------+-----------------+---------+--------+-----------------------+-------+-----------+------------+
-----------+------------+--------------------+--------+--------------------------+-------------------+
cc@osfn-cc:~/cassandra-demo$ |
```

1) Topology & High-Level Flow

- **Cluster:** 3× etcd containers (`etcd0`, `etcd1`, `etcd2`) via Docker Compose. Clients/benchmarks run from `etcd1` container using `etcdctl`.
- **Fault tool: Charybdefs** running on the **host** as a FUSE filesystem with a Thrift mgmt service (listens on **:9090**).
- **Mounting strategy:** Only the **leader's** data directory is placed on the Charybdefs mount. In our runs we pin the leader to **`etcd2`** and mount its data dir from **`/mnt/slowfs/etcd2`** (FUSE) backed by **`/data/raw/etcd2`** (real disk).
- **Results:** For every run we write CSVs to `~/cassandra-demo/results/<TIMESTAMP>_<label>/`:
  - `per_op_latency.csv` (per operation lat in seconds)
  - `latency_per_sec.csv` (avg p50/p95/p99 per second)
  - `throughput_per_sec.csv` (ops/sec time series)

Run flow (per scenario):

1. Verify cluster health → 2) Force leadership to `etcd2` → 3) Clear faults → 4) (Optionally) inject fs delay on the leader's path → 5) Run mini-benchmark and capture metrics → 6) Clear faults.

2) One-Time Prerequisites

- Docker + Docker Compose installed; `python3` available.

- Directories on host for Charybdefs mounts and real backing storage.
- `~/cassandra-demo` workspace with `docker-compose-etcd.yml` (3-node etcd) and our helper scripts.
- `~/charybdefs` built from source (provides the FUSE server that speaks Thrift on :9090 in our build).

3) Host Setup & Mounting (exact commands we used)

Use these when (re)starting or cleaning the FUSE mount and etcd cluster.

Stop the etcd cluster (ok if not running)

cd ~/cassandra-demo

docker compose -f docker-compose-etcd.yml down || true

 # Stop any previous Charybdefs and unmount the FUSE path

sudo pkill charybdefs || true

**sudo umount -l /mnt/slowfs/etcd2 2>/dev/null **

|| fusermount -u /mnt/slowfs/etcd2 2>/dev/null || true

```
cc@osfn-cc:~/charybdefs$ sudo pkill charybdefs || true
/mnt/slowfs/etcd2 2>/dev/null || true
cc@osfn-cc:~/charybdefs$ sudo umount -l /mnt/slowfs/etcd2 2>/dev/null \
>   || fusermount -u /mnt/slowfs/etcd2 2>/dev/null || true
cc@osfn-cc:~/charybdefs$ |
```

 # Prepare backing and mount directories

sudo mkdir -p /data/raw/etcd2 /mnt/slowfs/etcd2

sudo chown -R root:root /data/raw/etcd2 /mnt/slowfs/etcd2

sudo chmod 700 /data/raw/etcd2 /mnt/slowfs/etcd2

```
cc@osfn-cc:~/charybdefs$ sudo mkdir -p /data/raw/etcd2 /mnt/slowfs/etcd2
cc@osfn-cc:~/charybdefs$ sudo chown -R root:root /data/raw/etcd2 /mnt/slowfs/etcd2
hmod 700 /data/raw/etcd2 /mnt/slowfs/etcd2
cc@osfn-cc:~/charybdefs$ sudo chmod 700 /data/raw/etcd2 /mnt/slowfs/etcd2
cc@osfn-cc:~/charybdefs$ |
```

 # Start Charybdefs (FUSE). It listens on 9090; backs /mnt/slowfs/etcd2 to /data/raw/etcd2

**nohup sudo "$HOME/charybdefs/charybdefs" /mnt/slowfs/etcd2 **

**-omodules=subdir,subdir=/data/raw/etcd2 -oallow_other,nonempty **

> /tmp/charybdefs.log 2>&1 &

 # give FUSE a moment

sleep 2

 # Start etcd cluster

cd ~/cassandra-demo

 # Ensure docker-compose-etcd.yml binds etcd2's data dir to /mnt/slowfs/etcd2

 # (etcd0/1 may use fast local dirs)

docker compose -f docker-compose-etcd.yml up -d

 # For convenience during runs

export ENDPOINTS="http://etcd0:2379,http://etcd1:2379,http://etcd2:2379"

Checks:

- `tail -n 100 /tmp/charybdefs.log` (FUSE mounted, Thrift server ready)
- `docker compose ps` (all etcd containers up)
- `docker exec etcd1 etcdctl --endpoints="$ENDPOINTS" endpoint status -w table`

4) Files We Added / Modified

4.1 `charyb_fault.py` (Thrift client wrapper)

Small Python helper living in `~/cassandra-demo/charyb_fault.py` that talks to Charybdefs mgmt (host:port) and exposes **two** commands used by our shell orchestrator:

- `clear-all` — clears any active fs faults
- `set-delay --path <mount_path> --delay-us <μs>` — injects a WAL/write/fsync delay on the given mount path

> In our environment this script imports your generated Thrift stubs from `~/charybdefs/gen-py`. If you relocate stubs, set PYTHONPATH accordingly. We used these exact invocations from the orchestrator below.

4.2 `run_etcd_fsdelay.sh` (orchestrator; final working version)

Save as `~/cassandra-demo/run_etcd_fsdelay.sh` and `chmod +x` it.

D: > UChicago Internship > Reproducing-Filesystem-Delay-One-Size-Fits-None-NSDI25 > run_etcd_fsdelay.sh

```bash
#!/usr/bin/env bash
# run_etcd_fsdelay.sh
# Baseline vs WAL delay on the LEADER node's FUSE-mounted data dir.

set -euo pipefail

# ===== CONFIG DEFAULTS =====
OPS="${OPS:-200}"                         # number of put ops
MODE="${1:-baseline}"                     # baseline | delay
LEADER_TARGET="${LEADER_TARGET:-etcd2}"   # we want etcd2 as leader (the slow WAL)
WAL_DELAY_US="${WAL_DELAY_US:-0}"         # e.g. 0, 100000, 300000, 750000
RESULTS_DIR="${RESULTS_DIR:-results}"
OUT_PREFIX="${OUT_PREFIX:-etcd_fsdelay}"
ETCDCTL="${ETCDCTL:-/usr/local/bin/etcdctl}"
ETCD_CONTAINER="${ETCD_CONTAINER:-etcd0}" # container to exec etcdctl from
ENDPOINTS="${ENDPOINTS:-http://etcd0:2379,http://etcd1:2379,http://etcd2:2379}"
PYTHON="${PYTHON:-python3}"
CHARYB="${CHARYB:-./charyb_fault.py}"

# charybdefs control (your daemon listens here)
CHARYB_HOST="${CHARYB_HOST:-127.0.0.1}"
CHARYB_PORT="${CHARYB_PORT:-9090}"

# Methods and regex to hit etcd WAL on the slow node (matches your working setup)
WAL_METHODS="${WAL_METHODS:-open,create,write,write_buf,fsync,fdatasync,fsyncdir,flush}"
WAL_REGEX="${WAL_REGEX:-(^|.*/)member/wal/.*}"   # path as charybdefs sees it (starts with /)

# Optional: verify the delay is actually seen on the WAL path (host-side)
VERIFY_DELAY="${VERIFY_DELAY:-1}"

mkdir -p "$RESULTS_DIR"

echo "Target leader   : ${LEADER_TARGET:-'(current)'}"
echo "Mode            : $MODE"
echo "Ops             : $OPS"
[[ "$MODE" == "delay" ]] && echo "WAL delay (us)  : $WAL_DELAY_US"
```

```
C:\Users\jeezx>scp -i D:/key/yizzz-mj-trace.pem D:/Repo/code_sweepdelay/run_etcd_fsdelay.sh cc@192.5.86.216:/home/cc/cassandra-demo/
run_etcd_fsdelay.sh                                                                     100% 4802    15.2KB/s   00:00

C:\Users\jeezx>
```

Why these changes matter

- **Leader forcing**: `etcdctl move-leader` expects **decimal** member IDs; we parse JSON and pass the correct value to avoid the earlier hex/format errors.
- **Leader detection**: We treat the endpoint whose `Status.leader == Status.header.member_id` as the leader (stable in practice).
- **No `jq` dep**: We parse JSON with tiny Python one-liners so the script runs everywhere.
- **Fault mgmt**: We do **not** patch Charybdefs; we control it via Thrift using our Python wrapper and set `PYTHONPATH` to your `gen-py` stubs to prevent `ModuleNotFoundError` incidents.
- **Metrics**: We record per-op latency and derive **per-second** latency percentiles and throughput to match the figures you plotted (`per_op_latency.csv`, `latency_per_sec.csv`, `throughput_per_sec.csv`).

-

5) Running the Experiments

5.1 Baseline

cd ~/cassandra-demo

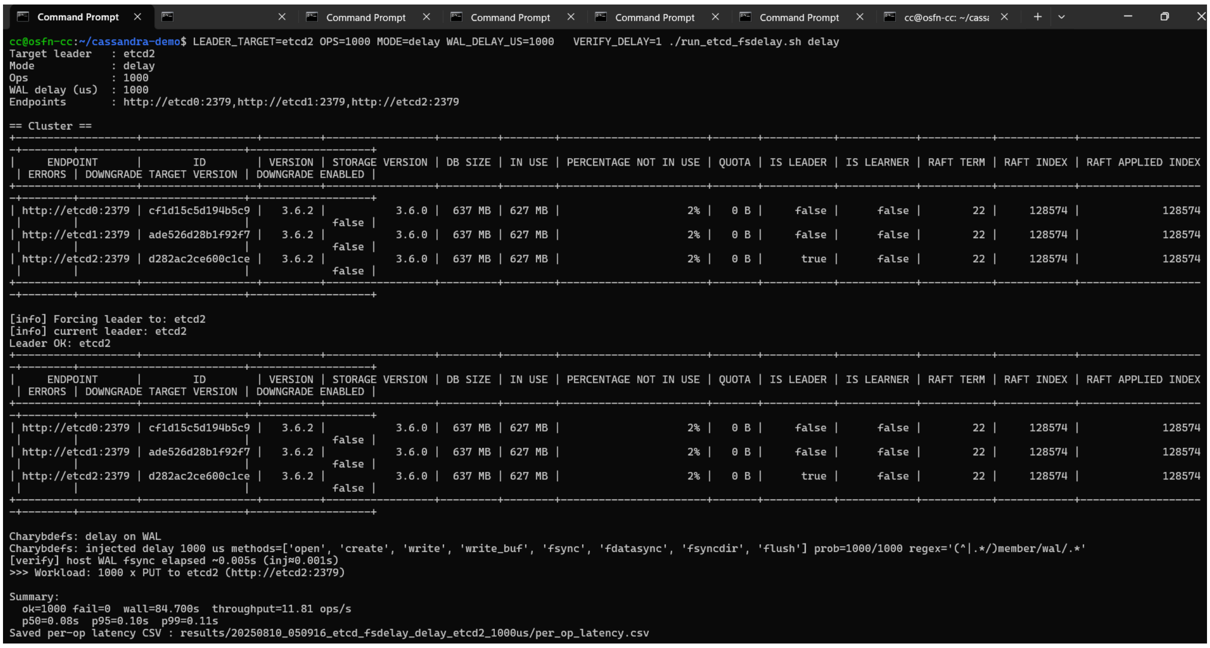
LEADER_TARGET=etcd2 OPS=200 ./run_etcd_fsdelay.sh baseline

This checks health, pins leadership to `etcd2`, clears faults, runs the benchmark, and writes CSVs under `results/<TIMESTAMP>_etcd_fsdelay_baseline_etcd2_0us/`.

5.2 Filesystem delay severities (1–100 ms)

> Our severity grid in **microseconds**: `1000 5000 10000 25000 50000 75000 100000` → **1, 5, 10, 25, 50, 75, 100 ms**

Run individually, e.g. **5 ms**:



```
cc@osfn-cc:~/cassandra-demo$ LEADER_TARGET=etcd2 OPS=1000 MODE=delay WAL_DELAY_US=1000   VERIFY_DELAY=1 ./run_etcd_fsdelay.sh delay
Target leader   : etcd2
Mode            : delay
Ops             : 1000
WAL delay (us)  : 1000
Endpoints       : http://etcd0:2379,http://etcd1:2379,http://etcd2:2379

== Cluster ==
+------------------+------------------+---------+-----------------+---------+--------+-----------------------+-------+-----------+------------+-----------+------------+--------------------
-+--------+-------------------------+-------------------+
|     ENDPOINT     |        ID        | VERSION | STORAGE VERSION | DB SIZE | IN USE | PERCENTAGE NOT IN USE | QUOTA | IS LEADER | IS LEARNER | RAFT TERM | RAFT INDEX | RAFT APPLIED INDEX
 | ERRORS | DOWNGRADE TARGET VERSION | DOWNGRADE ENABLED |
+------------------+------------------+---------+-----------------+---------+--------+-----------------------+-------+-----------+------------+-----------+------------+--------------------
-+--------+-------------------------+-------------------+
| http://etcd0:2379 | cf1d15c5d194b5c9 |   3.6.2 |           3.6.0 |  637 MB | 627 MB |                    2% |   0 B |     false |      false |        22 |     128574 |             128574
 |        |                          |             false |
| http://etcd1:2379 | ade526d28b1f92f7 |   3.6.2 |           3.6.0 |  637 MB | 627 MB |                    2% |   0 B |     false |      false |        22 |     128574 |             128574
 |        |                          |             false |
| http://etcd2:2379 | d282ac2ce600c1ce |   3.6.2 |           3.6.0 |  637 MB | 627 MB |                    2% |   0 B |      true |      false |        22 |     128574 |             128574
 |        |                          |             false |
+------------------+------------------+---------+-----------------+---------+--------+-----------------------+-------+-----------+------------+-----------+------------+--------------------
-+--------+-------------------------+-------------------+

[info] Forcing leader to: etcd2
[info] current leader: etcd2
Leader OK: etcd2
+------------------+------------------+---------+-----------------+---------+--------+-----------------------+-------+-----------+------------+-----------+------------+--------------------
-+--------+-------------------------+-------------------+
|     ENDPOINT     |        ID        | VERSION | STORAGE VERSION | DB SIZE | IN USE | PERCENTAGE NOT IN USE | QUOTA | IS LEADER | IS LEARNER | RAFT TERM | RAFT INDEX | RAFT APPLIED INDEX
 | ERRORS | DOWNGRADE TARGET VERSION | DOWNGRADE ENABLED |
+------------------+------------------+---------+-----------------+---------+--------+-----------------------+-------+-----------+------------+-----------+------------+--------------------
-+--------+-------------------------+-------------------+
| http://etcd0:2379 | cf1d15c5d194b5c9 |   3.6.2 |           3.6.0 |  637 MB | 627 MB |                    2% |   0 B |     false |      false |        22 |     128574 |             128574
 |        |                          |             false |
| http://etcd1:2379 | ade526d28b1f92f7 |   3.6.2 |           3.6.0 |  637 MB | 627 MB |                    2% |   0 B |     false |      false |        22 |     128574 |             128574
 |        |                          |             false |
| http://etcd2:2379 | d282ac2ce600c1ce |   3.6.2 |           3.6.0 |  637 MB | 627 MB |                    2% |   0 B |      true |      false |        22 |     128574 |             128574
 |        |                          |             false |
+------------------+------------------+---------+-----------------+---------+--------+-----------------------+-------+-----------+------------+-----------+------------+--------------------
-+--------+-------------------------+-------------------+

Charybdefs: delay on WAL
Charybdefs: injected delay 1000 us methods=['open', 'create', 'write', 'write_buf', 'fsync', 'fdatasync', 'fsyncdir', 'flush'] prob=1000/1000 regex='(^|.*/)member/wal/.*'
[verify] host WAL fsync elapsed ~0.005s (inj≈0.001s)
>>> Workload: 1000 x PUT to etcd2 (http://etcd2:2379)

Summary:
  ok=1000 fail=0  wall=84.700s  throughput=11.81 ops/s
  p50=0.08s  p95=0.10s  p99=0.11s
Saved per-op latency CSV : results/20250810_050916_etcd_fsdelay_delay_etcd2_1000us/per_op_latency.csv
```

LEADER_TARGET=etcd2 OPS=200 WAL_DELAY_US=5000 ./run_etcd_fsdelay.sh delay

Or sweep them all:

for us in 1000 5000 10000 25000 50000 75000 100000; do

  LEADER_TARGET=etcd2 OPS=200 WAL_DELAY_US=$us ./run_etcd_fsdelay.sh delay

done

Each run produces a new timestamped directory, e.g.:

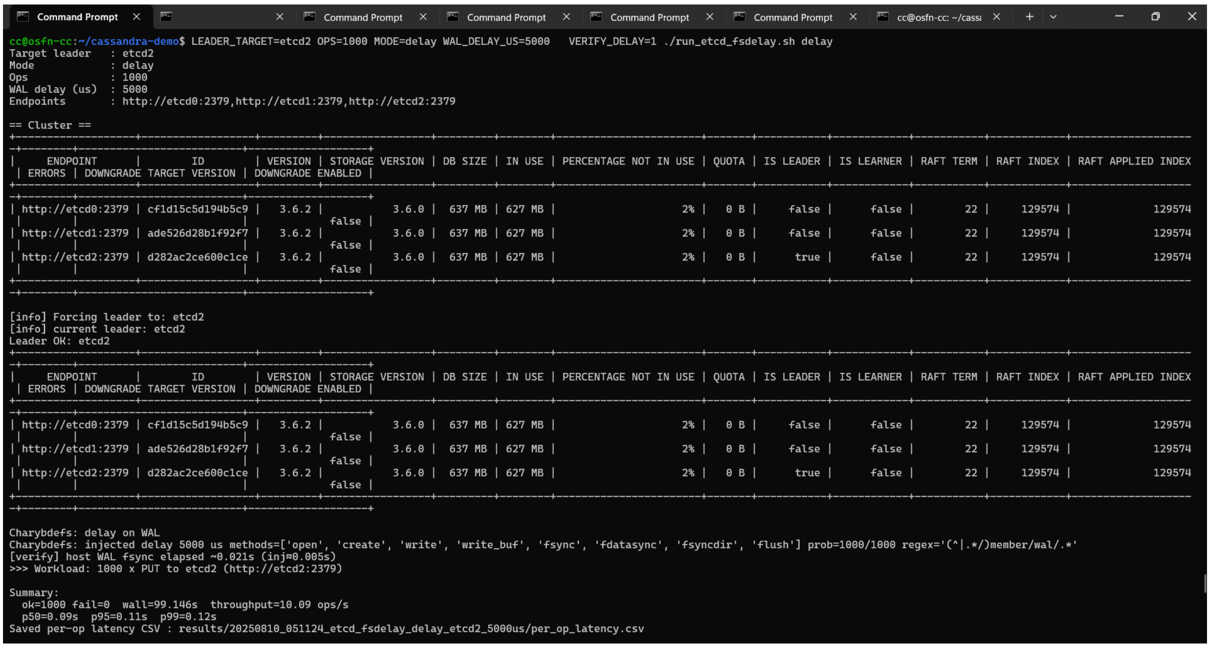
results/

20250809_222805_etcd_fsdelay_baseline_etcd2_0us/

per_op_latency.csv latency_per_sec.csv throughput_per_sec.csv

20250809_225811_etcd_fsdelay_delay_etcd2_100us/

20250810_051124_etcd_fsdelay_delay_etcd2_5000us/



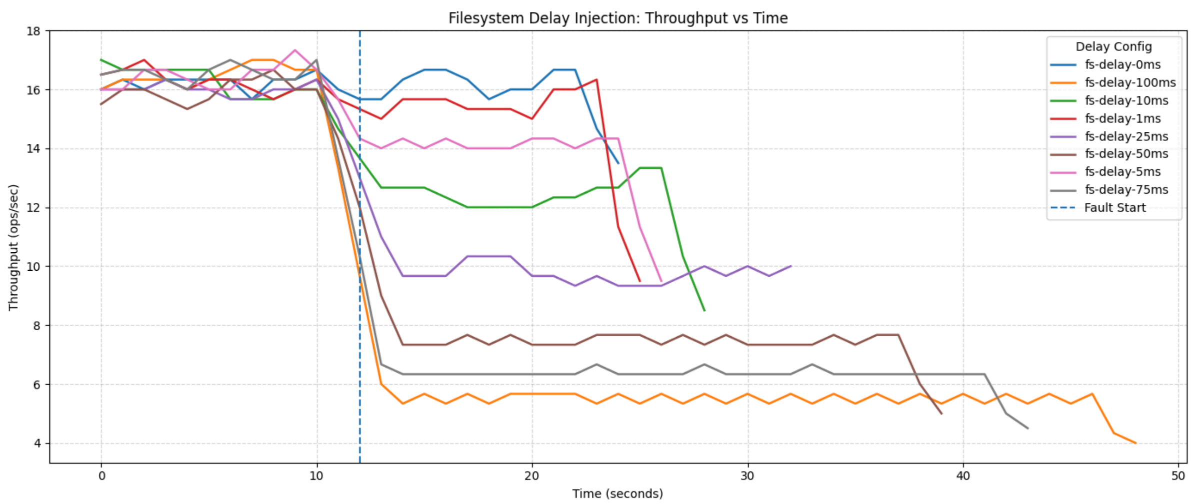
```
cc@osfn-cc:~/cassandra-demo$ LEADER_TARGET=etcd2 OPS=1000 MODE=delay WAL_DELAY_US=5000   VERIFY_DELAY=1 ./run_etcd_fsdelay.sh delay
Target leader   : etcd2
Mode            : delay
Ops             : 1000
WAL delay (us)  : 5000
Endpoints       : http://etcd0:2379,http://etcd1:2379,http://etcd2:2379

== Cluster ==
+------------------+------------------+---------+-----------------+---------+--------+-----------------------+--------+-----------+------------+-----------+------------+--------------------
-+--------+--------------------------+-------------------+
|     ENDPOINT     |        ID        | VERSION | STORAGE VERSION | DB SIZE | IN USE | PERCENTAGE NOT IN USE | QUOTA | IS LEADER | IS LEARNER | RAFT TERM | RAFT INDEX | RAFT APPLIED INDEX
 | ERRORS | DOWNGRADE TARGET VERSION | DOWNGRADE ENABLED |
+------------------+------------------+---------+-----------------+---------+--------+-----------------------+--------+-----------+------------+-----------+------------+--------------------
-+--------+--------------------------+-------------------+
| http://etcd0:2379 | cf1d15c5d194b5c9 |   3.6.2 |           3.6.0 |  637 MB | 627 MB |                    2% |   0 B |     false |      false |        22 |     129574 |             129574
 |        |                          |             false |
| http://etcd1:2379 | ade526d28b1f92f7 |   3.6.2 |           3.6.0 |  637 MB | 627 MB |                    2% |   0 B |     false |      false |        22 |     129574 |             129574
 |        |                          |             false |
| http://etcd2:2379 | d282ac2ce600c1ce |   3.6.2 |           3.6.0 |  637 MB | 627 MB |                    2% |   0 B |      true |      false |        22 |     129574 |             129574
 |        |                          |             false |
+------------------+------------------+---------+-----------------+---------+--------+-----------------------+--------+-----------+------------+-----------+------------+--------------------
-+--------+--------------------------+-------------------+

[info] Forcing leader to: etcd2
[info] current leader: etcd2
Leader OK: etcd2
+------------------+------------------+---------+-----------------+---------+--------+-----------------------+--------+-----------+------------+-----------+------------+--------------------
-+--------+--------------------------+-------------------+
|     ENDPOINT     |        ID        | VERSION | STORAGE VERSION | DB SIZE | IN USE | PERCENTAGE NOT IN USE | QUOTA | IS LEADER | IS LEARNER | RAFT TERM | RAFT INDEX | RAFT APPLIED INDEX
 | ERRORS | DOWNGRADE TARGET VERSION | DOWNGRADE ENABLED |
+------------------+------------------+---------+-----------------+---------+--------+-----------------------+--------+-----------+------------+-----------+------------+--------------------
-+--------+--------------------------+-------------------+
| http://etcd0:2379 | cf1d15c5d194b5c9 |   3.6.2 |           3.6.0 |  637 MB | 627 MB |                    2% |   0 B |     false |      false |        22 |     129574 |             129574
 |        |                          |             false |
| http://etcd1:2379 | ade526d28b1f92f7 |   3.6.2 |           3.6.0 |  637 MB | 627 MB |                    2% |   0 B |     false |      false |        22 |     129574 |             129574
 |        |                          |             false |
| http://etcd2:2379 | d282ac2ce600c1ce |   3.6.2 |           3.6.0 |  637 MB | 627 MB |                    2% |   0 B |      true |      false |        22 |     129574 |             129574
 |        |                          |             false |
+------------------+------------------+---------+-----------------+---------+--------+-----------------------+--------+-----------+------------+-----------+------------+--------------------
-+--------+--------------------------+-------------------+

Charybdefs: delay on WAL
Charybdefs: injected delay 5000 us methods=['open', 'create', 'write', 'write_buf', 'fsync', 'fdatasync', 'fsyncdir', 'flush'] prob=1000/1000 regex='(^|.*/)member/wal/.*'
[verify] host WAL fsync elapsed ~0.021s (inj≈0.005s)
>>> Workload: 1000 x PUT to etcd2 (http://etcd2:2379)

Summary:
  ok=1000 fail=0  wall=99.146s  throughput=10.09 ops/s
  p50=0.09s  p95=0.11s  p99=0.12s
Saved per-op latency CSV : results/20250810_051124_etcd_fsdelay_delay_etcd2_5000us/per_op_latency.csv
```

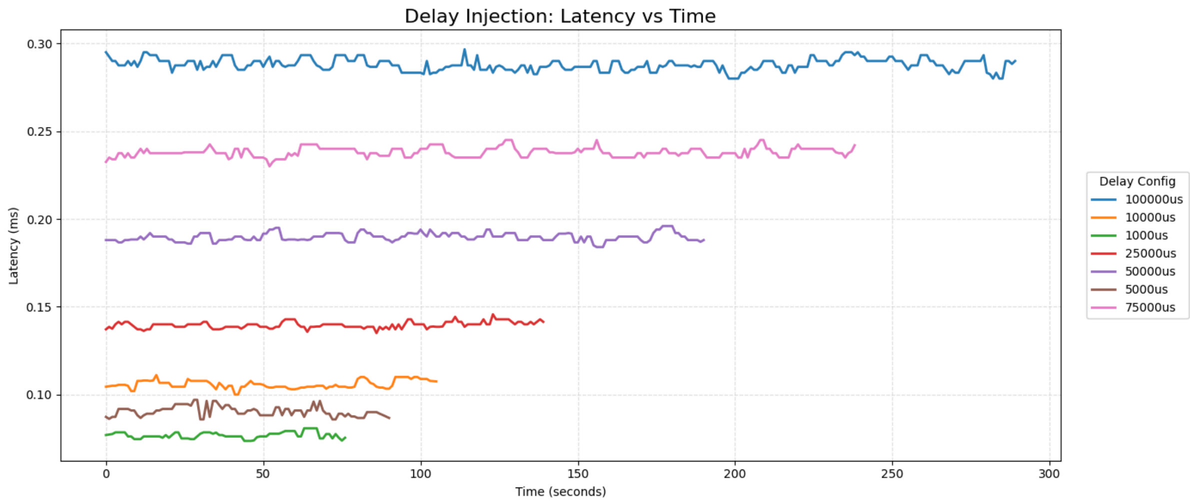
```
cc@osfn-cc:~$ cd cassandra-demo
cc@osfn-cc:~/cassandra-demo$ cd results/20250810_050916_etcd_fsdelay_delay_etcd2_1000us/
cc@osfn-cc:~/cassandra-demo/results/20250810_050916_etcd_fsdelay_delay_etcd2_1000us$ ls
latency_per_sec.csv  per_op_latency.csv  throughput_per_sec.csv
cc@osfn-cc:~/cassandra-demo/results/20250810_050916_etcd_fsdelay_delay_etcd2_1000us$ cd ../..
cc@osfn-cc:~/cassandra-demo$ cd results/20250810_051124_etcd_fsdelay_delay_etcd2_5000us
cc@osfn-cc:~/cassandra-demo/results/20250810_051124_etcd_fsdelay_delay_etcd2_5000us$ ls
latency_per_sec.csv  per_op_latency.csv  throughput_per_sec.csv
cc@osfn-cc:~/cassandra-demo/results/20250810_051124_etcd_fsdelay_delay_etcd2_5000us$ cd ../..
cc@osfn-cc:~/cassandra-demo$ cd results/20250810_051315_etcd_fsdelay_delay_etcd2_10000us
cc@osfn-cc:~/cassandra-demo/results/20250810_051315_etcd_fsdelay_delay_etcd2_10000us$ ls
latency_per_sec.csv  per_op_latency.csv  throughput_per_sec.csv
cc@osfn-cc:~/cassandra-demo/results/20250810_051315_etcd_fsdelay_delay_etcd2_10000us$ cd ../..
cc@osfn-cc:~/cassandra-demo$ cd  results/20250810_051546_etcd_fsdelay_delay_etcd2_25000us
cc@osfn-cc:~/cassandra-demo/results/20250810_051546_etcd_fsdelay_delay_etcd2_25000us$ ls
latency_per_sec.csv  per_op_latency.csv  throughput_per_sec.csv
cc@osfn-cc:~/cassandra-demo/results/20250810_051546_etcd_fsdelay_delay_etcd2_25000us$ cd ../..
cc@osfn-cc:~/cassandra-demo$ cd  results/20250810_051818_etcd_fsdelay_delay_etcd2_50000us
cc@osfn-cc:~/cassandra-demo/results/20250810_051818_etcd_fsdelay_delay_etcd2_50000us$ ls
latency_per_sec.csv  per_op_latency.csv  throughput_per_sec.csv
cc@osfn-cc:~/cassandra-demo/results/20250810_051818_etcd_fsdelay_delay_etcd2_50000us$ cd ../..
cc@osfn-cc:~/cassandra-demo$ cd results/20250810_052230_etcd_fsdelay_delay_etcd2_75000us
cc@osfn-cc:~/cassandra-demo/results/20250810_052230_etcd_fsdelay_delay_etcd2_75000us$ ls
latency_per_sec.csv  per_op_latency.csv  throughput_per_sec.csv
cc@osfn-cc:~/cassandra-demo/results/20250810_052230_etcd_fsdelay_delay_etcd2_75000us$ cd ../..
cc@osfn-cc:~/cassandra-demo$ cd  results/20250810_052751_etcd_fsdelay_delay_etcd2_100000us
cc@osfn-cc:~/cassandra-demo/results/20250810_052751_etcd_fsdelay_delay_etcd2_100000us$ ls
latency_per_sec.csv  per_op_latency.csv  throughput_per_sec.csv
```

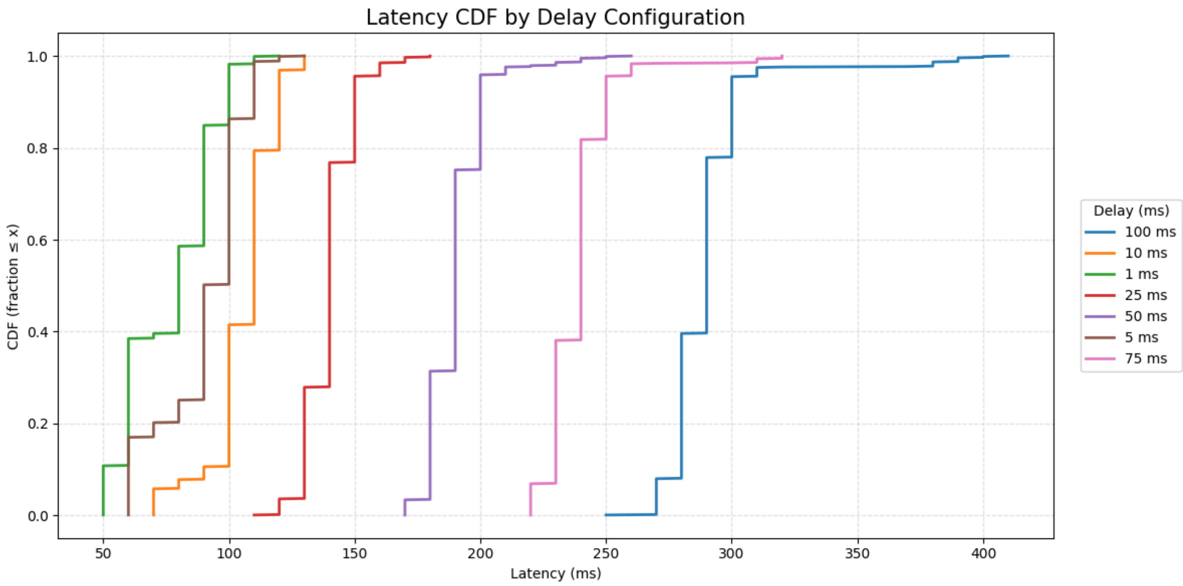
6) Reading & Using the Outputs

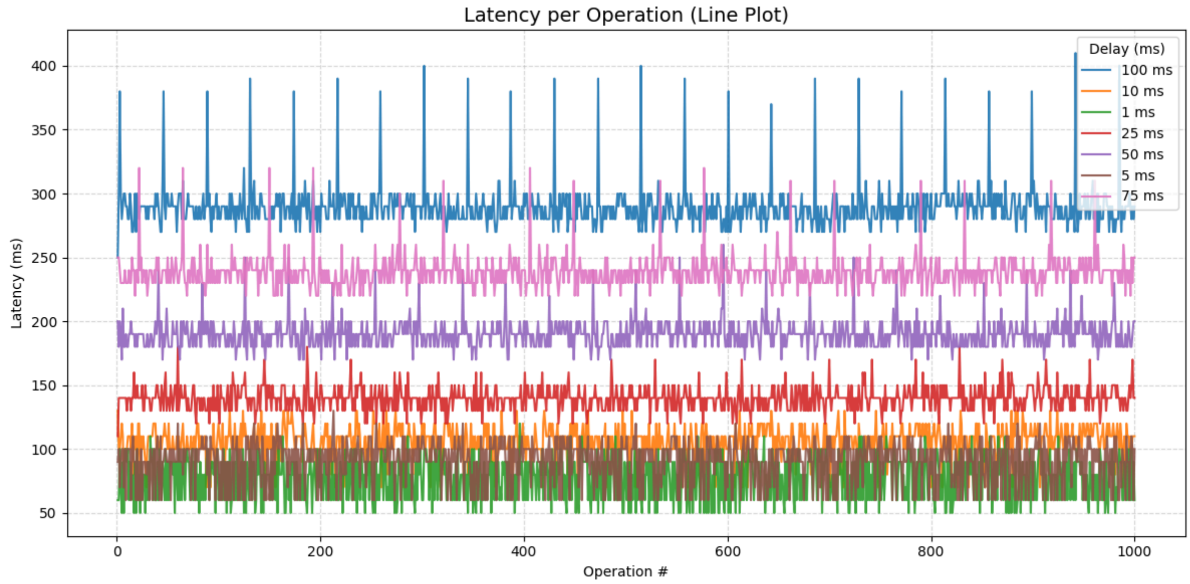
- **Per-op latency:** `per_op_latency.csv` with `op,seconds`. Useful for line/CDF plots across delays.
- **Latency per second:** `latency_per_sec.csv` with `second,p50_ms,p95_ms,p99_ms`. Good for seeing steady-state vs. spikes.
- **Throughput per second:** `throughput_per_sec.csv` with `second,ops`. Good for identifying “collapse zones”.

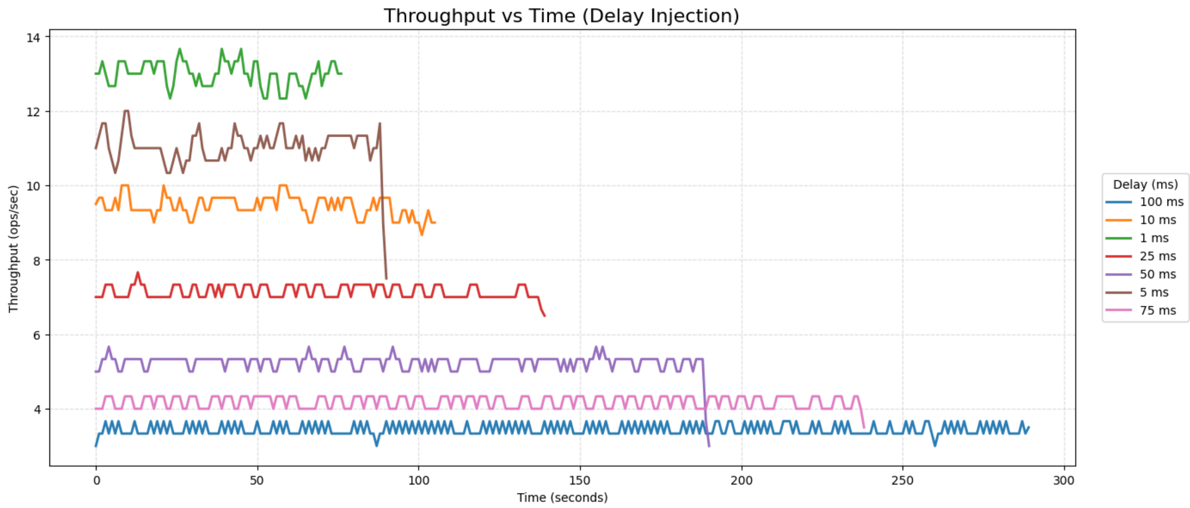
Our plot scripts (line charts, CDFs, and latency vs. throughput overlays) consume these CSVs directly. The figures you generated for the paper reproduction came from these files.




Delay Injection: Latency vs Time
Latency (ms)
Time (seconds)
Delay Config
100000us
10000us
1000us
25000us
50000us
5000us
75000us


Latency CDF by Delay Configuration
CDF (fraction ≤ x)
Latency (ms)
Delay (ms)
100 ms
10 ms
1 ms
25 ms
50 ms
5 ms
75 ms





7) Validation & Sanity Checks

- **Leader really on etcd2?**
  docker exec etcd1 etcdctl --endpoints="$ENDPOINTS" endpoint status -w table

Charybdefs active?
mount | grep slowfs

tail -n 50 /tmp/charybdefs.log

- ss -lntp | grep :9090   # mgmt listening
- **After a delay run, latency shifts** should be visible in `per_op_latency.csv` and lower `throughput_per_sec.csv` relative to baseline. If not, re-check that etcd2's data dir in Compose is bound to `/mnt/slowfs/etcd2` (not a fast path) and that leadership is actually on etcd2 **during** the run.

8) Troubleshooting (issues we actually hit)

- **`move-leader` complaining about invalid ID** → We were passing hex; script now parses decimal.
- **`ModuleNotFoundError` for Thrift stubs** → Ensure `PYTHONPATH=$HOME/charybdefs/gen-py` before calling the Python client.
- **FUSE unmount busy** → Use `umount -l ... || fusermount -u ...` as shown; then restart Charybdefs.
- **No noticeable impact across small delays** → Verify the leader is on the **slowfs**-**backed** node during the **entire** run; confirm delay values are **microseconds** not milliseconds; verify ops load (`OPS`) is enough to surface the effect.

9) Quick Command Recap

 # One-time (or after a reset)

cd ~/cassandra-demo

docker compose -f docker-compose-etcd.yml down || true

sudo pkill charybdefs || true

sudo umount -l /mnt/slowfs/etcd2 2>/dev/null || fusermount -u /mnt/slowfs/etcd2 2>/dev/null || true

sudo mkdir -p /data/raw/etcd2 /mnt/slowfs/etcd2

sudo chown -R root:root /data/raw/etcd2 /mnt/slowfs/etcd2

sudo chmod 700 /data/raw/etcd2 /mnt/slowfs/etcd2

**nohup sudo "$HOME/charybdefs/charybdefs" /mnt/slowfs/etcd2 **

**-omodules=subdir,subdir=/data/raw/etcd2 -oallow_other,nonempty **

> /tmp/charybdefs.log 2>&1 &

docker compose -f docker-compose-etcd.yml up -d

export ENDPOINTS="http://etcd0:2379,http://etcd1:2379,http://etcd2:2379"

 # Baseline

LEADER_TARGET=etcd2 OPS=200 ./run_etcd_fsdelay.sh baseline

 # Delays (1–100 ms)

for us in 1000 5000 10000 25000 50000 75000 100000; do

```
  LEADER_TARGET=etcd2 OPS=200 WAL_DELAY_US=$us
./run_etcd_fsdelay.sh delay

done
```

10) Notes Mapping Back to the Paper

- **Fault type:** filesystem delay on the leader's log/write path (via Charybdefs), matching the paper's fs-slow methodology.
- **Severity grid:** 1–100 ms (we used 1, 5, 10, 25, 50, 75, 100 ms).
- **Metric emphasis:** throughput degradation alongside p95/p99 latency (tail latency alone can be misleading for varying durations).
- **Pipeline:** init → warm-up (if needed) → inject → measure → clear (our runs keep them short; add warm-up as desired).

Appendix A — Environment Variables (for quick tuning)

Variable	Meaning	Default
`ENDPOINTS`	etcd endpoints CSV	`http://etcd0:2379,http://etcd1:2379,http://etcd2:2379`
`LEADER_TARGET`	node to hold leadership	`etcd2`
`OPS`	ops per run (if `DURATION_SEC=0`)	`200`
`DURATION_SEC`	time-based run if >0	`0`
`WAL_DELAY_US`	injected fs delay in microseconds	`0`

OUTROOT	results root dir	$PWD/results
MOUNT_PREFIX	Charybdefs mount root on host	/mnt/slowfs
CHARY_HOST/CHARY_PORT	mgmt endpoint	127.0.0.1 / 9090
CHARY_CLI	path to our Thrift Python wrapper	$PWD/charyb_fault.py