Contents

1 Benchmarking OpenFGA

This document summarize the benchmark of LXD querying performance in regards to OpenFGA. The setup is as follows:

- Create a 3 node cluster
- Create 10 projects (`project[1..10]`)
- In each project create one profile `bench-profile` and one volume `bench-volume`
- In each project, create one instance with one profile `bench-profile`

Then the benchmark does:

- A listing of all the project with recursion with an `admin`, `restricted` and `fine-grained` client. We do it 50 times to get an average result and a standard deviation.
- If the version of LXD supports attaching entitlements to the returned projects, we also use the `fine-grained` client to call the API 50 times with the `with-entitlements=operator` query parameter.
- We do these calls on the `leader` node and on a `non-leader` node to observe the discrepancies generated by the forwarded API calls. The result returned by the `leader` is usually faster.

1.1 `latest/edge`

(The HEAD is at: `e49b74882a: VM: Handle not found file descriptor (#14479)` at the time of this benchmark)

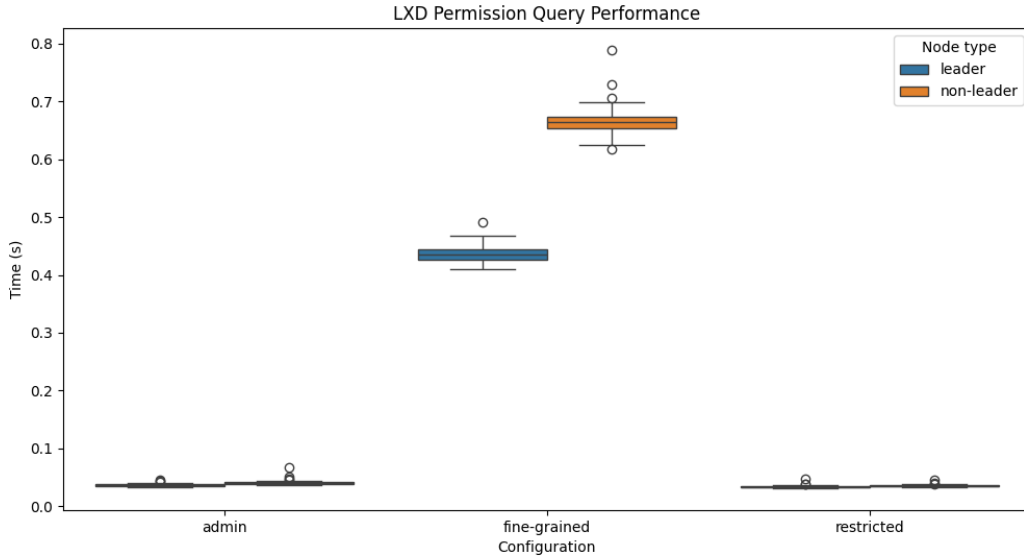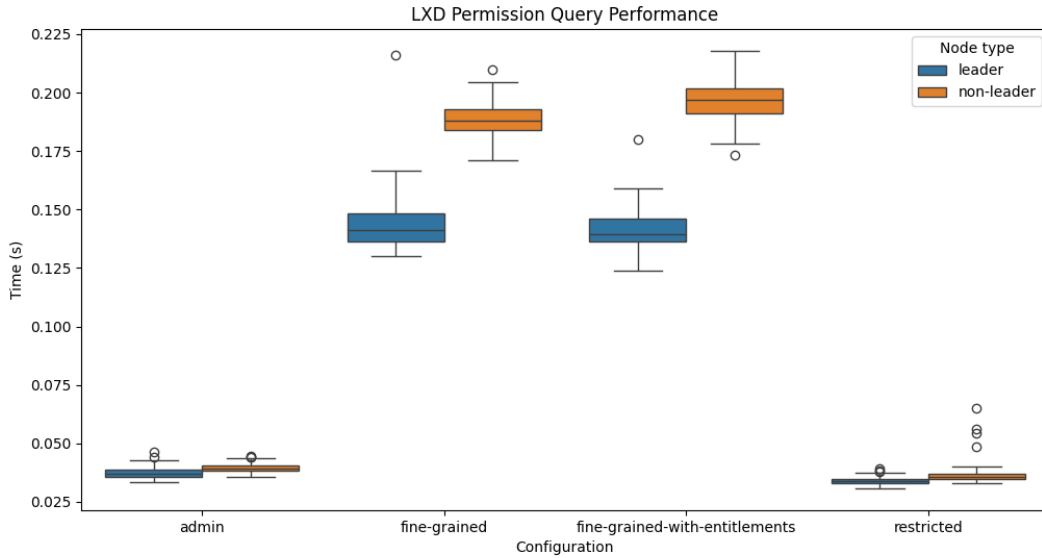| Case	Mean (ms)	Median (ms)	Std dev (ms)	Min (ms)	Max (ms)
fine-grained-with-entitlements+leader	NaN	NaN	NaN	NaN	NaN
fine-grained-with-entitlements+non-leader	NaN	NaN	NaN	NaN	NaN
fine-grained+leader	438	435	15	411	491
fine-grained+non-leader	667	664	26	617	789
restricted+leader	34	34	2	31	48
restricted+non-leader	35	35	1	33	45
admin+leader	37	36	2	34	46
admin+non-leader	41	40	4	36	67

Figure 1: latest_edge_bench

1.2 Our work: with/without `&with-entitlements=operator`, no concurrency in datastore.

(Note: we won't be detail the `restricted` and `admin` results as they were unaffected.)

Case	Mean (ms)	Median (ms)	Std dev (ms)	Min (ms)	Max (ms)
fine-grained-with-entitlements+leader	142	139	9	124	180
fine-grained-with-entitlements+non-leader	197	197	9	173	218
fine-grained+leader	144	141	13	130	216
fine-grained+non-leader	188	188	7	171	210

1.3 Our work: with/without `&with-entitlements=operator`, with concurrency in datastore.
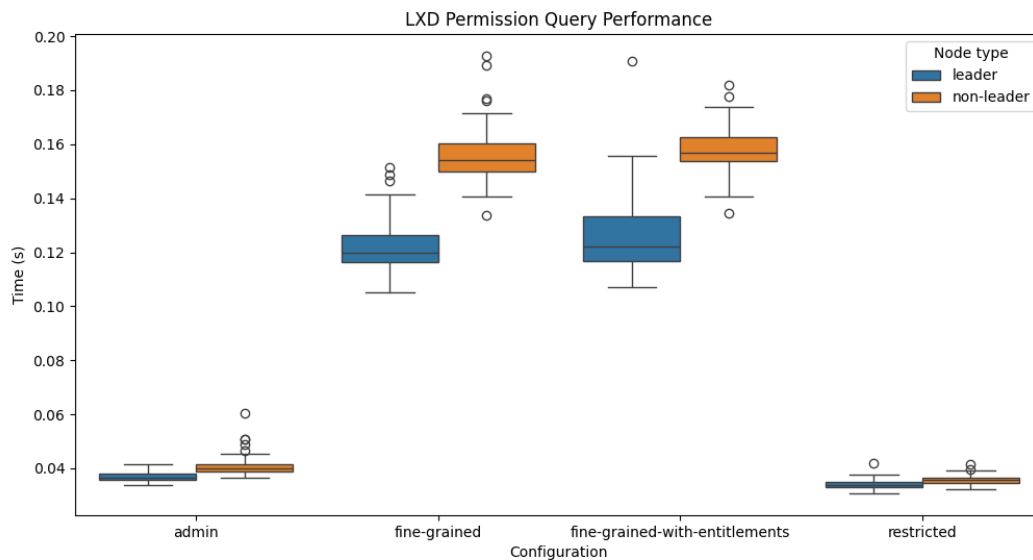

Figure 2: our_work_one_entitlement_concurrency

Case	Mean (ms)	Median (ms)	Std dev (ms)	Min (ms)	Max (ms)
fine-grained-with-entitlements+leader	126	122	14	107	190
fine-grained-with-entitlements+non-leader	158	156	9	134	181
fine-grained+leader	122	119	10	105	151
fine-grained+non-leader	156	154	11	134	192

1.4 Our work: with/without `&with-entitlements=all`, with concurrency in datastore.

Case	Mean (ms)	Median (ms)	Std dev (ms)	Min (ms)	Max (ms)
fine-grained-with-entitlements+leader	240	237	14	219	283
fine-grained-with-entitlements+non-leader	331	329	18	290	398
fine-grained+leader	123	119	10	104	163
fine-grained+non-leader	157	156	19	132	274

1.5 Based on @markylaing PR (https://github.com/canonical/lxd/pull/14513), with/without `&with-entitlements=operator`, with concurrency in datastore.

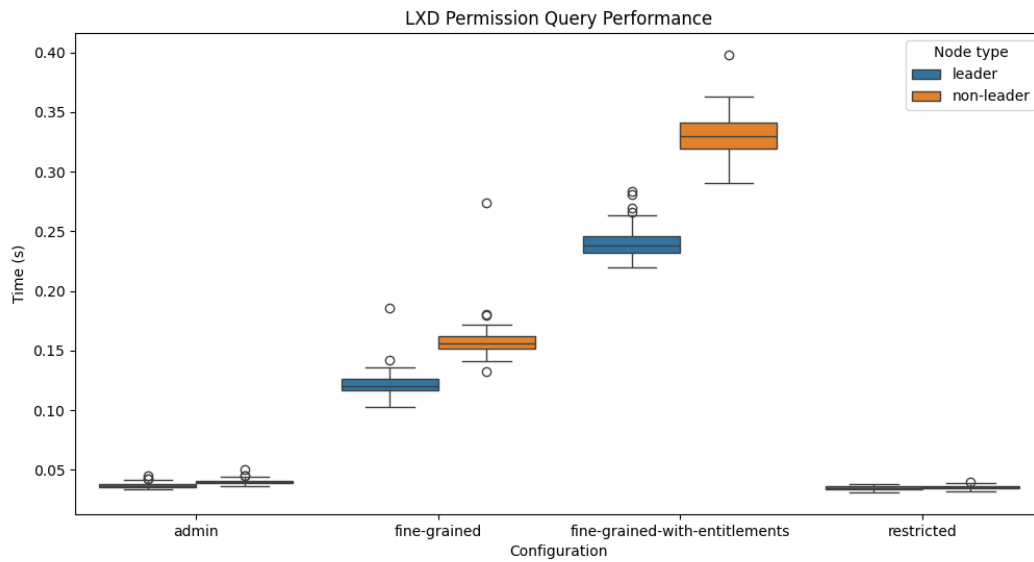| Case	Mean (ms)	Median (ms)	Std dev (ms)	Min (ms)	Max (ms)
fine-grained-with-entitlements+leader	161	156	2	143	232
fine-grained-with-entitlements+non-leader	204	202	10	186	240
fine-grained+leader	159	157	12	138	206
fine-grained+non-leader	202	199	16	180	291

Figure 3: our_work_all_compatible_entitlements_concurrency
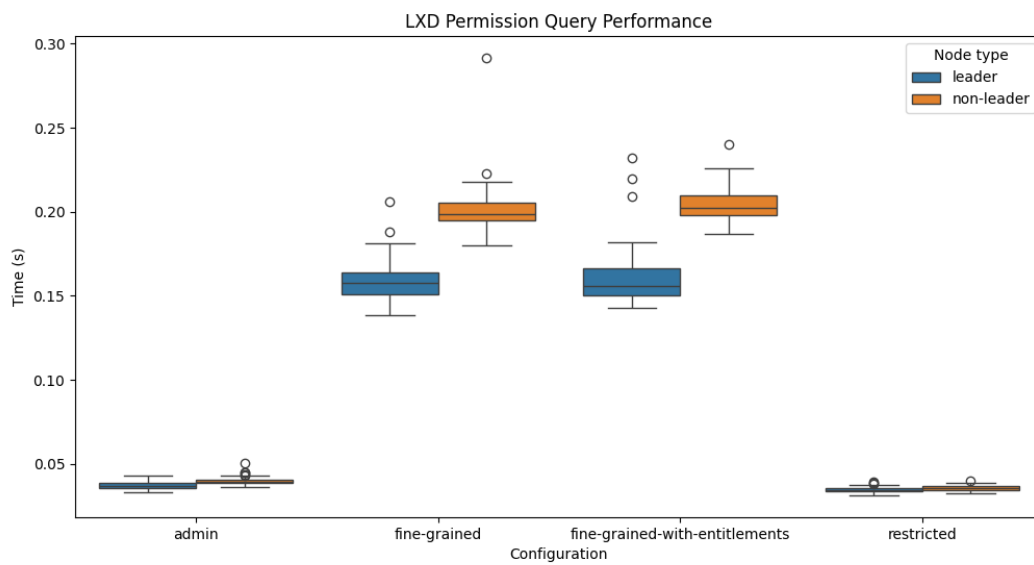
Figure 4: markylaing_work_one_entitlement_concurrency

1.6 Based on @markylaing PR (https://github.com/canonical/lxd/pull/14513), with/without `&with-entitlements=all`, with concurrency in datastore.
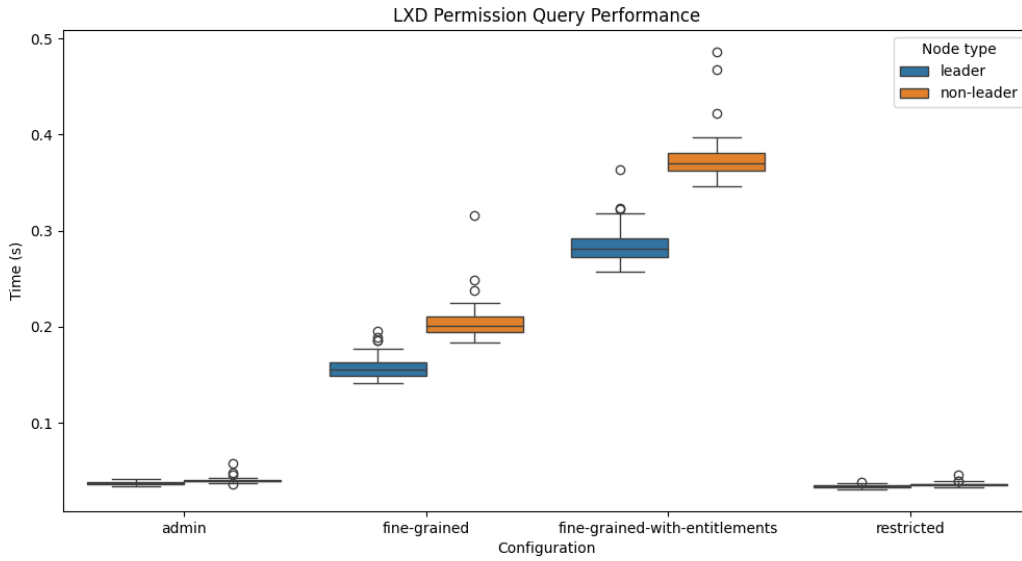

Figure 5: markylaing__work_all_compatible_entitlements_concurrency

Case	Mean (ms)	Median (ms)	Std dev (ms)	Min (ms)	Max (ms)
fine-grained-with-entitlements+leader	285	281	19	258	363
fine-grained-with-entitlements+non-leader	375	370	25	345	486
fine-grained+leader	158	156	12	141	195
fine-grained+non-leader	205	201	20	183	315