10 Pitfalls on The Path to Osquery Bliss

Zach Wasserman — Osquery/Fleet Consultant, Dactiv LLC

QueryCon 2019

Pitfall #1

User context is important when executing queries

User context is important when executing queries

- As user:
  SELECT * FROM firefox_addons;
- As root:
  SELECT * FROM firefox_addons;

User context is important when executing queries

- Osquery sometimes uses the user context in which it is running to retrieve results.

- **Solution:** JOIN with the users table.
  SELECT * **FROM** users
  JOIN firefox_addons **USING** (uid);

Pitfall #2

Order of JOINed tables can be significant

Order of JOINed tables can be significant

- As root:
  SELECT * FROM firefox_addons
  JOIN users **USING** (uid);

Order of JOINed tables can be significant

- The order in which the tables are generated can effect the constraints the generation function receives.

- **Solution:** Order the JOINs so that tables that require constraints are generated after.

```
SELECT * FROM users
JOIN firefox_addons USING (uid);
```

- Note: Sometimes the SQLite optimizer will reorder the tables anyway. To be sure the tables are JOINed in the order provided, use CROSS JOIN.

```
SELECT * FROM users
CROSS JOIN firefox_addons USING (uid)
WHERE identifier LIKE '%mozilla%';
```

and breathe

Pitfall #3

Dude, where's my SHA1?
Reading large files and the --read_max flag

Reading large files and the --read_max flag

- **SELECT * FROM** hash
  WHERE path = '/Users/zwass/suspicious';

Reading large files and the --read_max flag

- Tables that try to read files over the --read_max size (default 50MB) can return empty results.
- This can effect most tables and osquery functions that involve reading files, not just the hash table!
- **Solution:** Tune the --read_max flag if you need results from large files.

Pitfall #4

JSON Escaping and Query Packs

JSON Escaping and Query Packs

- Let's copy a query from the windows-attacks query pack:

```
...
  "CCleaner_Trojan.Floxif": {
      "query" : "select * from registry where path like 'HKEY_LOCAL_MACHINE\\SOFTWARE\\Piriform\\Agomo%';",
...
```

JSON Escaping and Query Packs

- JSON backslashes are escaped as '\\', while osqueryi expects backslashes to use the literal '\'.

- **Solution:** Be careful to use the appropriate escaping and modify for the format when translating between osqueryi and JSON query packs.

- Note: The fleetctl format uses yaml and therefore does not require any escaping in backslashes. This means that queries can be directly copy/pasted to osqueryi.

Pitfall #5

CLI Flags vs. Configuration Options

CLI Flags vs. Configuration Options

- Let's try setting the extensions_socket configuration in our config file:

```
{
 "options": {
   "extensions_socket": "/tmp/osquery_ext.sock"
  }
}
```

CLI Flags vs. Configuration Options

- Some options must be specified as CLI flags (and can't be modified after osquery startup), while others are configurable in a loaded configuration.
- osqueryd --help will tell us which flags are CLI-only
- **Solution:** Identify flags that are CLI-only and specify those in explicit flags or a flagfile.

Pitfall #6

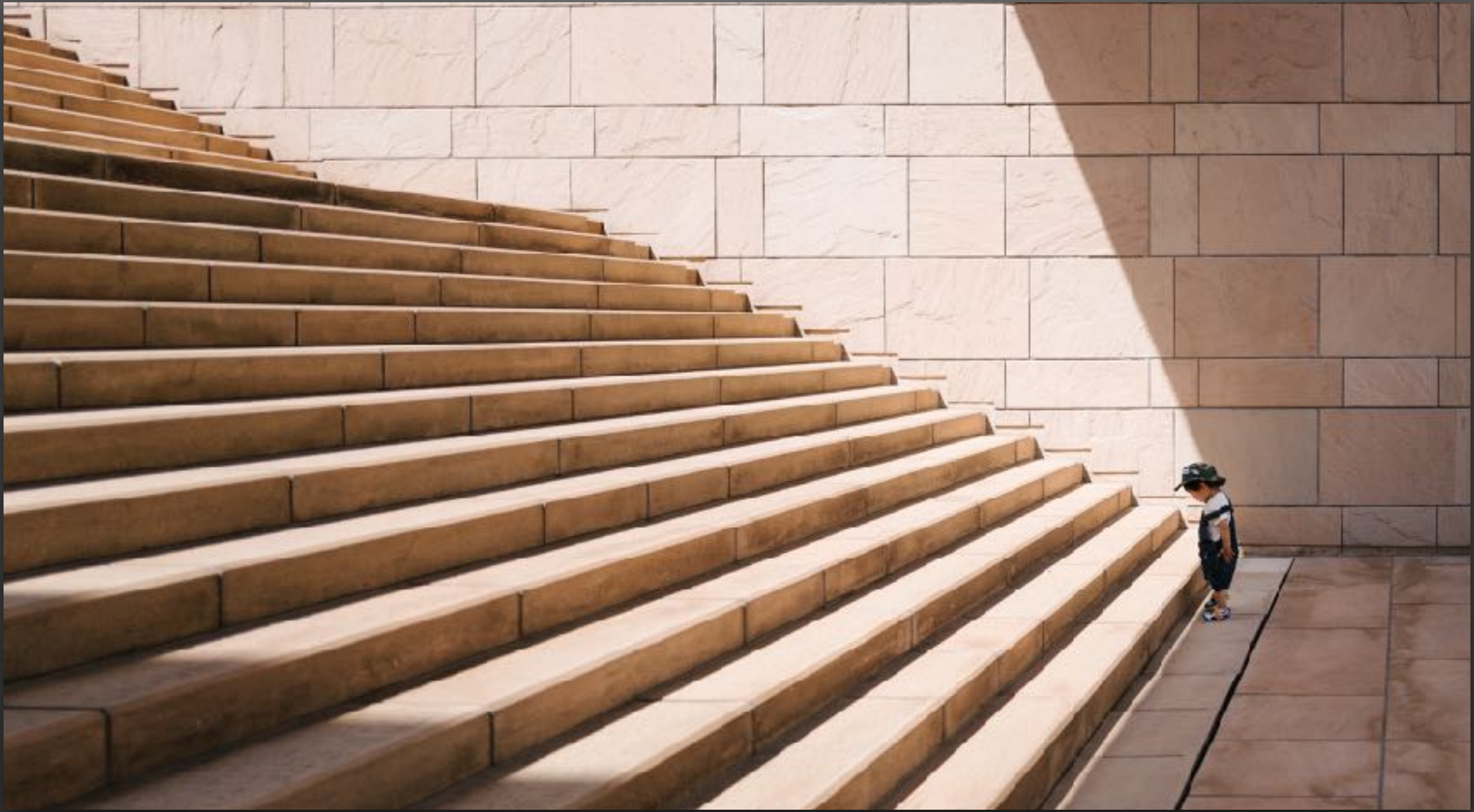
Understanding schedule intervals

Understanding schedule intervals

- Schedule a query.
- Put the computer to sleep.
- When does the query run?

Understanding schedule intervals

- The osquery scheduler runs on ticks (while the process is active), not wall time.
- **Solution:** Account for time the machine is off or suspended when creating query intervals.



Pitfall #7

Events in osqueryd and osqueryi

Events in osqueryd and osqueryi

- Run osqueryd and see that events are collected.
- Run osqueryi and query for the events.
  Where are they?

Events in osqueryd and osqueryi

- An ephemeral database is used with osqueryi by default.
- **Solution:** Provide the --database_path flag to osqueryi to open the RocksDB database used by osqueryd.
- Note: Only one osquery process can open a database at a time. Terminate osqueryd before connecting osqueryi to the database.

Pitfall #8

Tuning event expiration flags

Tuning event expiration flags

- Run osquery with a low events_max:

```
{
 "options": {
   "disable_events": false,
   "events_max": 4
  }
}
```

Tuning event expiration flags

- The flags --events_max and --events_expiration prevent the events buffers from growing indefinitely.

- **Solution:** Ensure that the flags are tuned appropriately for the query intervals and volumes of data being generated by event publishers.

Pitfall #9

Event publisher status

Event publisher status

- osqueryd is running with events enabled
- How can we understand why events are not coming through publishers?

Event publisher status

- The osquery_events tables provides status information about event publishers and subscribers
- **Solution:** Look at the active, events, and subscriptions columns of the osquery_events table for the relevant publishers.
  SELECT * FROM osquery_events;

Pitfall #10

Identifying expensive queries

Identifying expensive queries

- With osqueryd running a schedule
- How can we identify which queries are utilizing the most resources?

Identifying expensive queries

- The osquery_schedule table exposes metadata about the scheduled queries and their resource consumption.

- **Solution:** Look for outliers in the osquery_schedule table
  SELECT * **FROM** osquery_schedule
  ORDER BY user_time + system_time **DESC**

- Note: The osquery repository also has performance tooling at /tools/analysis/profile.py.

YOU'VE
GOT
THIS

Zach Wasserman
github.com/zwass
Osquery Slack: @zwass
Twitter: @thezachw
zach@dactiv.llc