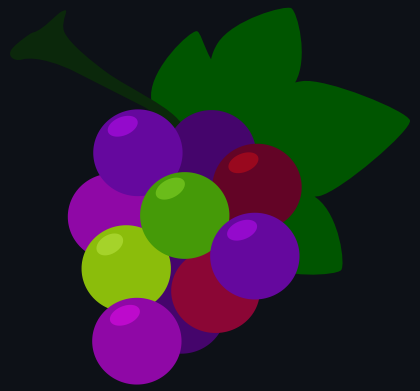
AWS Deployment Load Test



Setup

- Using Locust for load generation
- Tasks:
  - *well-known*: get provisioning service info

```
GET .well-known/veraison/provisioning
```

  - *provision*: provision a PSA corim

```
POST /endorsement-porvisioning/v1/submit
```

  - *verify*: verify PSA evidence

```
POST /challenge-response/v1/newSession?nonce=[NONCE]
POST /challenge-response/v1/[SESSION_ID]
DELETE /challenge-response/v1/[SESSION_ID]
```

- Measure requests/s (throughput) and 95th percentile for response time (latency)
- The same CoRIM/token used for all requests

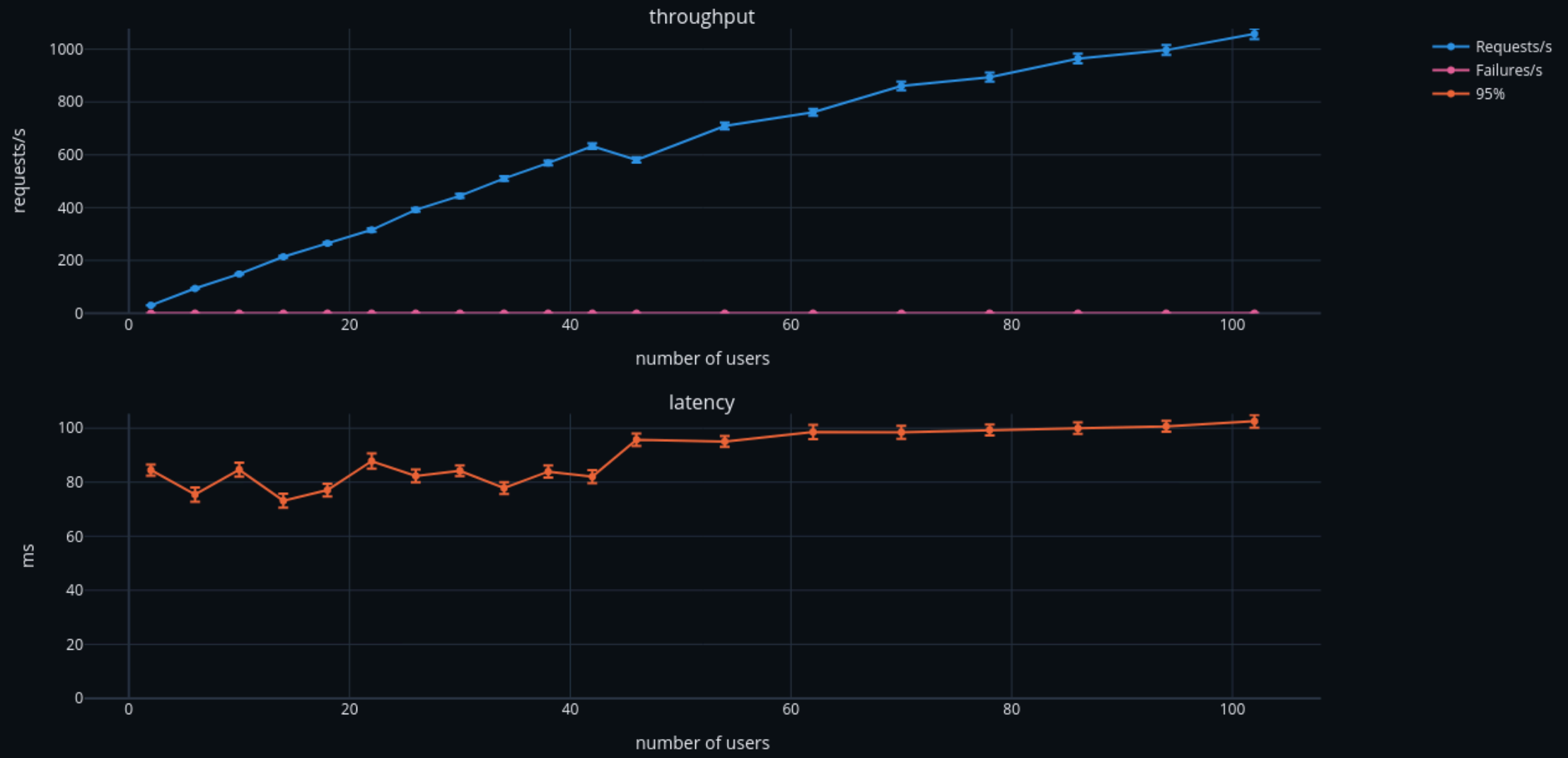
Setup

- `t2.micro` EC2 instances
- `db.t3.micro` RDS instance running PostgreSQL
- `cache.t2.micro` ElastiCache Memcached instance (used for verification session management)
- Deployment estimated monthly cost: ~$160

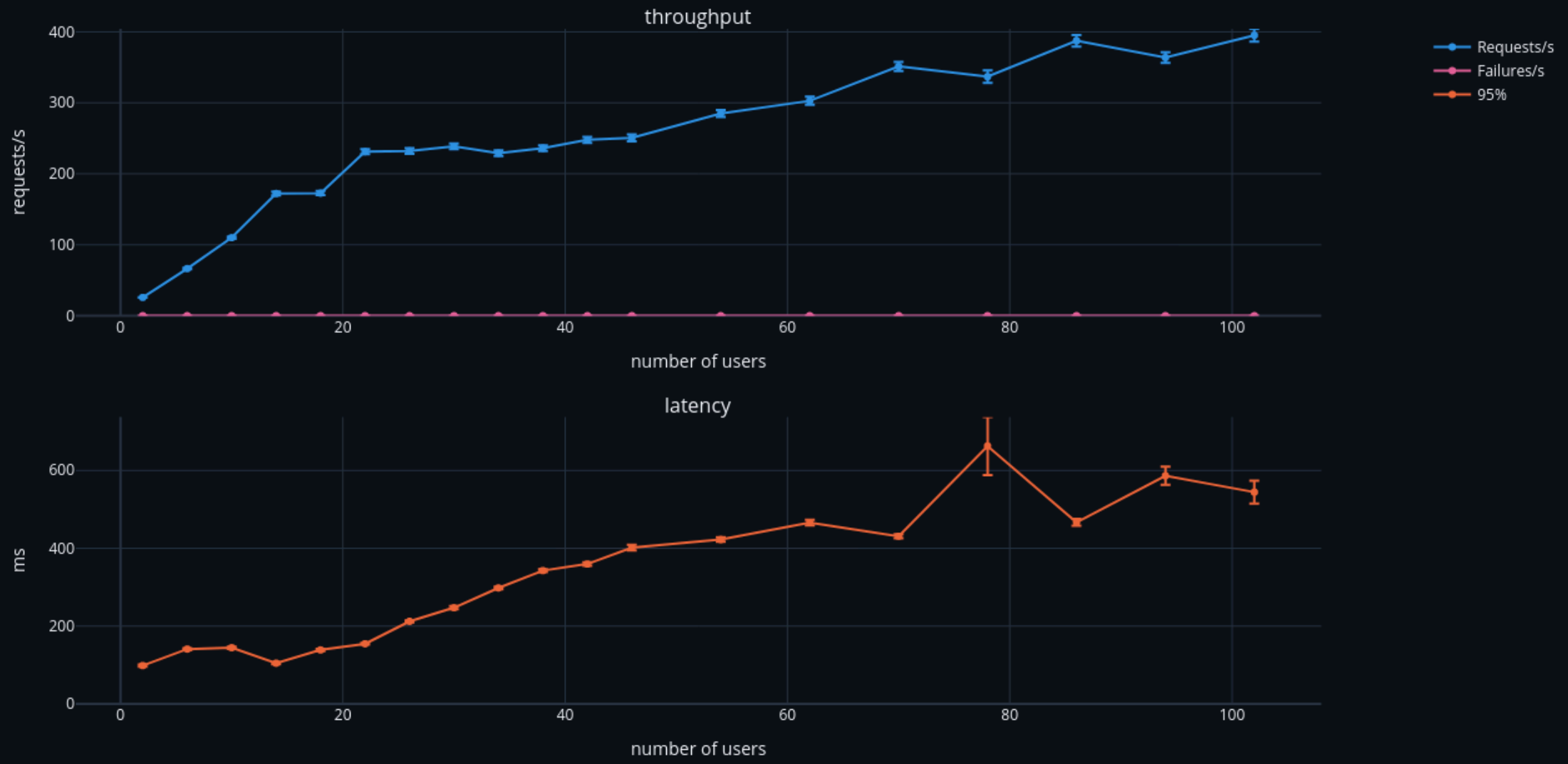
Trial 1

- Loop all three tasks repeatedly
- Spawn progressively large number of users with each set of runs
- Clear stores between each run
- Individual run duration: 150s
- Auto-scaling groups for provisioning and verification services pinned to 3 instances

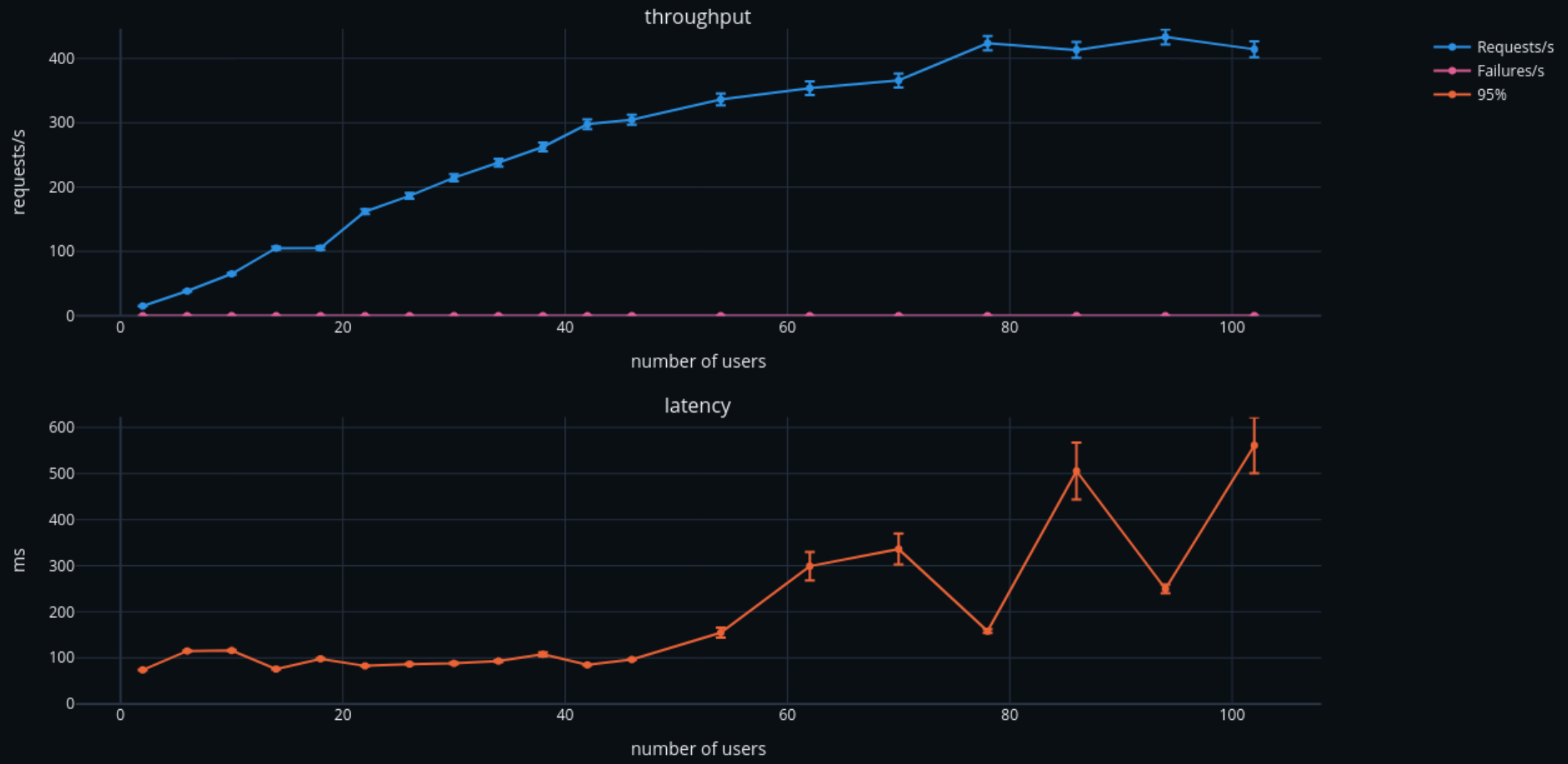
Requests against AWS (3 prov, 3 verif) **GET /.well-known/veraison/provisioning**



Requests against AWS (3 prov, 3 verif) **POST /endorsement-provisioning/v1/submit**



Requests against AWS (3 prov, 3 verif) **POST /challenge-response/v1/[SESSION_ID]**



Timelines

Observations

- Throughput seems to flatten out at just over 400 requests/s
- A large initial spike in latency for higher user counts
- A small number of failed requests at higher user counts during verification

users	method	location	failures	total requests
86	POST	/challenge-response/v1/[SESSION_ID]	36	42081
102	POST	/challenge-response/v1/[SESSION_ID]	31	42343

```
ERROR   vts     failed to connect to `user=veraison database=veraison`:
        10.1.3.77:5432 (veraison-support-rdsinstance-jlqj0tz7na5h.cdp1uev75r4z.eu-west-1.rds.amazonaws.com): server error: FATAL: sorry, too many clients already (SQLSTATE 53300)
        10.1.3.77:5432 (veraison-support-rdsinstance-jlqj0tz7na5h.cdp1uev75r4z.eu-west-1.rds.amazonaws.com): server error: FATAL: sorry, too many clients already (SQLSTATE 53300)
ERROR   vts     failed to connect to `user=veraison database=veraison`:
        10.1.3.77:5432 (veraison-support-rdsinstance-jlqj0tz7na5h.cdp1uev75r4z.eu-west-1.rds.amazonaws.com): server error: FATAL: remaining connection slots are reserved for roles with the SUPERUSER attribute (SQLSTATE 53300)
        10.1.3.77:5432 (veraison-support-rdsinstance-jlqj0tz7na5h.cdp1uev75r4z.eu-west-1.rds.amazonaws.com): server error: FATAL: no pg_hba.conf entry for host "10.1.2.198", user "veraison", database "veraison", no encryption (SQLSTATE 28000)
```

Trial 2

- Measure the impact of additional services instances
- Same as *Trial 1*, except
- Auto-scaling groups for provisioning and verification services pinned to 1 instance

3p3v vs. 1p1v **GET /.well-known/veraison/provisioning**

3p3v vs. 1p1v **POST /endorsement-provisioning/v1/submit**

3p3v vs. 1p1v **POST /challenge-response/v1/[SESSION_ID]**

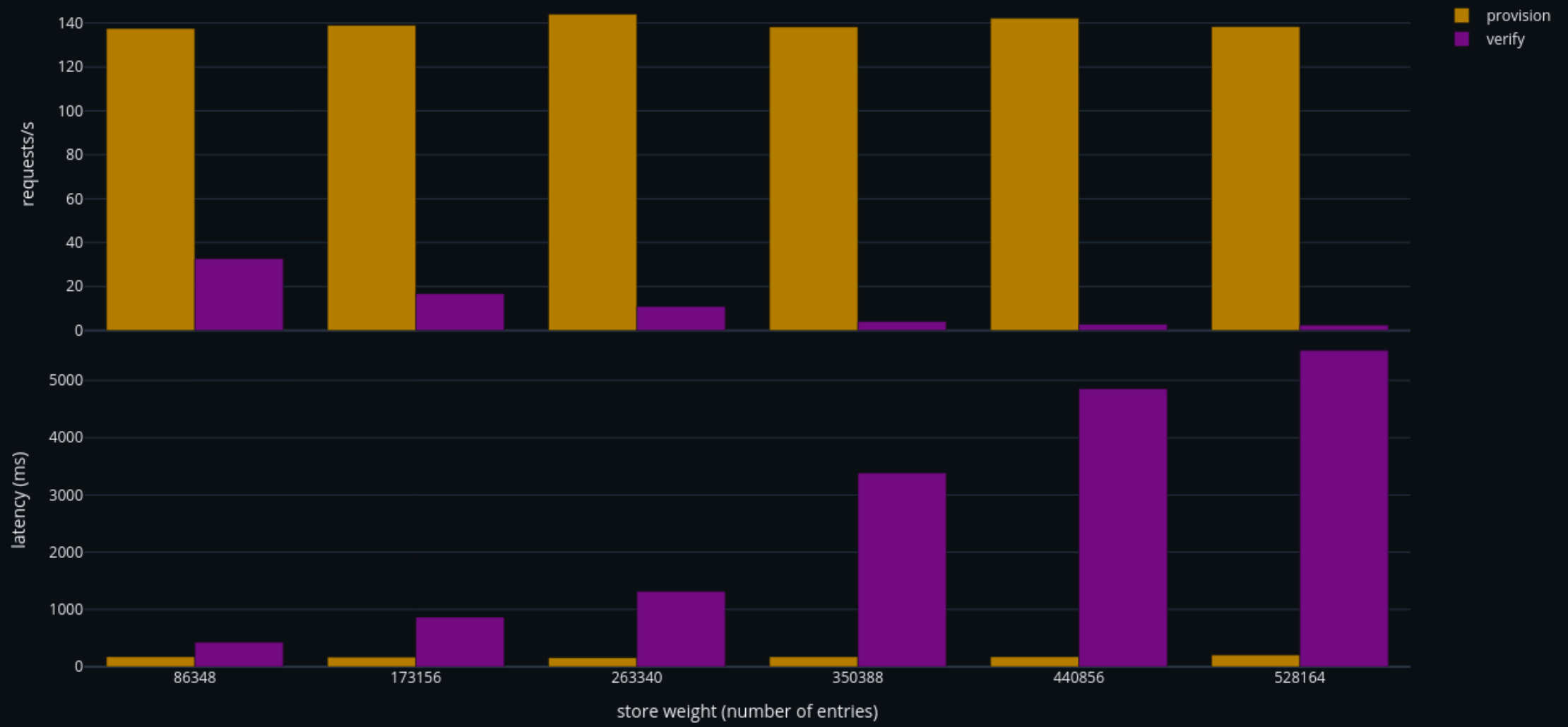
Observations

- Additional instances have a positive impact on throughput
  - Especially pronounced in verification (close to double at higher user counts)
- Additional instances have a *negative* impact on latency
  - Should be noted, this is at 95th percentile. Median (i.e. "expected" latency hovers around 100ms in all cases)

Trial 3

- Measure the impact of the number of records in the store (its "weight")
- Parallel users fixed at 15
- Run `provision` task followed by `verify` task
- For 6 iterations
- Number of entries in the K-V store (endorsements + trust anchors) is taken at the end of each iteration
- Stores are *NOT* cleared between tasks or between iterations
- Individual run duration: 150s
- Auto-scaling groups for provisioning and verification services pinned to 3 instances

Performance vs. K-V store weight



Observations

- Store weight has no impact on provisioning
- *Severe* impact on verification

Trial 4

- Evaluates auto-scaling
- A singe run with
  - 50 users
  - 900 second (15 minute) duration
  - Just the `verify` task
  - Auto-scaling between 1 and 3 instances

- New instances take >5mins to spin up
- There is a failed requests spike as an instance comes online

```
2025-03-03 09:35:33.709000+00:00 i-0a1ec1087f1abe77a ERROR      api-handler     rpc error: code = Unavailable desc = connection error: desc = "transport: Error while dialing: dial tcp 127.0.0.1:50051: connect: connection refused"
```

Conclusion

- Need a health check indicator -- could use .well-known?
  - This is not currently possible for provisioning due to auth.
- DB seems to be the bottleneck (especially for verification)
  - Implementing a cache may help?
  - Duplicate detection for provisioning?
  - Archiving for the K-V store?

Thank You