Pushing test lab to its limits

Paweł Wieczorek

FOSDEM’24
Agenda

- Background
- Interactive approaches
- Automation solutions
- Data generation
Background
Continuously growing a lab

Growing a lab for automated upstream testing, Laura Nao
Unusually high load

- No reason to panic – allocated resources are in use
- Highest on the nodes running database processes
Or is it?

Available SoCs

The results shown here cover the last 14 days of available data starting from Fri, 02 Feb 2024 (time is UTC based).

<table>
<thead>
<tr>
<th>SoC</th>
<th>Total Unique Labs</th>
<th>Total Unique Boards</th>
<th>Total Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>allwinner</td>
<td>7</td>
<td>21</td>
<td>Q</td>
</tr>
<tr>
<td>amlogic</td>
<td>6</td>
<td>10</td>
<td>Q</td>
</tr>
<tr>
<td>arc</td>
<td>1</td>
<td>1</td>
<td>Q</td>
</tr>
<tr>
<td>ai01</td>
<td>3</td>
<td>4</td>
<td>Q</td>
</tr>
<tr>
<td>broadcom</td>
<td>2</td>
<td>2</td>
<td>Q</td>
</tr>
<tr>
<td>davnnc</td>
<td>1</td>
<td>1</td>
<td>Q</td>
</tr>
<tr>
<td>exynos</td>
<td>1</td>
<td>1</td>
<td>Q</td>
</tr>
<tr>
<td>freescale</td>
<td>7</td>
<td>11</td>
<td>Q</td>
</tr>
</tbody>
</table>
Throw in more resources
LAVA architecture

Master

- Web interface
- Database
- lava-master daemon
- Scheduler

Worker

- LAVA-slave daemon
- Dispatcher
- Device Under Test

Device Under Test
LAVA under the hood

Master

Web interface → Database

lava-master daemon → Scheduler

Worker

LAVA-slave daemon → Dispatcher

Device Under Test
Interactive approaches
When in doubt – check log out
Django Debug Toolbar
Local instances

How to provide production workload?

- Initially often a clean slate
- Only virtual devices?
- Populating database with fixtures?
Model for lava-server
Two groups of people

case "$f" in
  *.sh)
    # https://github.com/docker-library/postgres/issues/450#issuecomment-393167936
    # https://github.com/docker-library/postgres/pull/452
    if [ -x "$f" ]; then
      printf "%s: running %s
" "$f"
      "$f"
    else
      printf "%s: sourcing %s
" "$f"
    fi
  fi
  ;;
  *.*sql)
    printf "%s: running %s
" "$f"; docker_process_sql -f "$f"; printf '
' ;;

  *.*sql.gz)
    printf "%s: running %s
" "$f"; gunzip -c "$f" | docker_process_sql; printf '
' ;;

  *.*sql.xz)
    printf "%s: running %s
" "$f"; xzcat "$f" | docker_process_sql; printf '
' ;;

  *.*sql.zst)
    printf "%s: running %s
" "$f"; zstd -dc "$f" | docker_process_sql; printf '
' ;;

  *)
    printf "%s: ignoring %s
" "$f" ;;
  esac
| Rank | Row Count | Duration | Disk I/O | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches | Index Cond | Heap Fetches |
|------|----------|----------|----------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 1    | 182,291  | 0.007    | 1.0       | 26          | 1           | 0           | Nested Loop | (cost=47.101.48 .72 rows=26 width=2.654) (actual time=181.321.182.298 rows=26 loops=1) Buffers: shared hit=349,167 read=3,564        |
| 2    | 182,291  | 0.035    | 1.0       | 26          | 1           | 0           | Nested Loop | (cost=47.101.48 .256.326.99 rows=12.273 width=2.054) (actual time=181.319.182.291 rows=26 loops=1) Buffers: shared hit=349,167 read=3,564        |
| 3    | 182,291  | 0.014    | 1.0       | 26          | 1           | 0           | Nested Loop | (cost=47.101.34 .254.266.69 rows=12.273 width=2.532) (actual time=181.310.182.230 rows=26 loops=1) Buffers: shared hit=349,119 read=3,564        |
| 4    | 182,164  | 0.017    | 1.0       | 26          | 1           | 0           | Nested Loop, Left Join | (cost=47.101.34 .254.106.46 rows=12.273 width=2.601) (actual time=181.296.182.164 rows=26 loops=1) Buffers: shared hit=349,112 read=3,564        |
| 5    | 182,095  | 41.194   |          | 26          | 1           | 0           | Index Scan | using test_index on index | (cost=47.101.19 .252.093.58 rows=12.273 width=1.962) (actual time=181.285.182.095 rows=26 loops=1) Index Cond: (requested_device_type_id IS NOT NULL) Filter: (is_public AND NOT (hashedSubPlan 1)) OR (hashedSubPlan 4) Buffers: shared hit=349,064 read=3,564        |
| 6    | 140.476  |          | 1.0       | 114,311     | 1           | 0           | Gather | (cost=1,000.85 .46.772.63 rows=114,211 width=4) (actual time=2.365.140.476 rows=114,211 loops=1) Workers Planned: 2 Workers Launched: 2 Buffers: shared hit=349,015 read=3,564        |
| 8    | 134,352  | 13.395   | 1.3       | 114,110     | 3           | / 3         | Nested Loop | (cost=47.85.34.351.53 rows=47,588 width=4) (actual time=2.475.134.352 rows=38,070 loops=3) Buffers: shared hit=349,015 read=3,564        |
| 9    | 6,768    | 6,768    | 1.3       | 114,211     | 3           | / 3         | Parallel Index Only Scan | using test_index on index | (cost=0.42 .2,600.86 rows=47,588 width=4) (actual time=0.425.6,748 rows=38,070 loops=3) Index Cond: (requested_device_id IS NOT NULL) Heap Fetches: 0 Buffers: shared hit=11 read=307        |
| 10   | 114,211  | 114,211  |          | 3           | 114,211     | / 3         | Index Only Scan | using test_index on index | (cost=43.0.67 rows=1 width=4) (actual time=0.003.0.003 rows=1 loops=1 rows=114,211) Index Cond: (id = $1.testJobId) Heap Fetches: 0 Buffers: shared hit=349,004 read=3,257        |
Automated tracking
Add your first benchmark to CI

How to fit benchmarks in existing tests?

- Cache warmup
- Calibration
- Result comparison
- Compatible framework
LAVA-compatible fixture

--- test session starts ---

platform: linux -- Python 3.9.2, pytest-6.0.2, py-1.10.0, pluggy-0.13.0 -- /usr/bin/python3
cachedir: .pytest_cache
benchmark: 3.2.2 (defaults: timer=time.perf_counter disable_gc=False min_rounds=5 min_time=0.000005 max_time=1.0 calibration_precision=10 warmup=False warmup_iterations=100000)
Django settings: lava_server.settings.dev (from ini file)
rootdir: /home/vagrant/lava, configfile: pytest.ini
plugins: cov-2.10.1, benchmark-3.2.2, django-3.5.1, mock-1.10.4
collected 1 item

tests/lava_rest_app/perf/test_api_perf2.py::TestRestApi::test_testjobs PASSED [100%]

--- benchmark: 1 tests ---

<table>
<thead>
<tr>
<th>Name (time in ms)</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>StdDev</th>
<th>Median</th>
<th>IQR</th>
<th>Outliers</th>
<th>OPS</th>
<th>Rounds</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_testjobs</td>
<td>442.3801</td>
<td>450.1671</td>
<td>446.2986</td>
<td>3.0554</td>
<td>445.8224</td>
<td>4.6830</td>
<td>2.0</td>
<td>2.407</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

Legend:
Outliers: 1 Standard Deviation from Mean; 1.5 IQR (InterQuartile Range) from 1st Quartile and 3rd Quartile.
OPS: Operations Per Second, computed as 1 / Mean

--- 1 passed in 5.15s ---

https://pytest-benchmark.readthedocs.io
Plug into GitLab CI pipeline

- **test**
  - test
  - benchmark
- **build**
  - build debian source
  - build docker image
- **upload**
  - upload to obs
- **OBS**
  - obs
- **deploy**
  - deploy staging
- **Downstream**
  - obs

- analyze
  - black
  - code_quality
  - codespell
  - coverage
  - dockerfiles
  - flawfinder-sast
  - pylint
  - schemas
  - semgrep-sast

- build
  - debian/11
  - debian/12
  - doc
  - docker-arm64-dispatcher
  - docker-arm64-server
Dedicated GitLab CI runner

https://docs.gitlab.com/runner/install/
Cache CI data resources

What data could be used here?

- Quicker feedback loop
- Mechanism already in place: https://gitlab.com/lava/ci-images
- Copy interactive approach almost 1:1
Data generation
class DeviceFactory(factory.django.DjangoModelFactory):
    class Meta:
        model = Device
        django_get_or_create = ("hostname",)

    hostname = factory.Faker("hostname", levels=0)
    device_type = factory.fuzzy.FuzzyChoice(DeviceType.objects.all())
    worker_host = factory.fuzzy.FuzzyChoice(Worker.objects.all())

@factory.post_generation
def create_device_template(
    self, create, create_device_template: bool = False, **kwargs
):
    if (not create) or (not create_device_template):
        return

    from pathlib import Path
    from django.conf import settings
    device_template_dir = Path(settings.DEVICES_PATH)

    with open(device_template_dir / (self.hostname + "_jinja2"), mode="w+t") as f:
        f.write(r"%s + f"extends '{self.device_type.name}.jinja2'" + r"%s")

https://gitlab.collabora.com/lava/lava/-/blob/collabora/production/lava_db_generator/
Bonus: data retention

- Should LAVA archive all the jobs?
- Can it be delegated?
- Retention mechanism available upstream
- Enabled in Helm chart

Is all this data really necessary?
Summary
Final thoughts

- Process, not a one-time action
- Frequent revisiting and adjustments
- Small changes can bring huge boosts
Thank you!
Images used

- https://www.freeimages.com/photo/burning-computer-1508147 by dknudsen
- https://pulsgdanska.pl/artykul/rzut-dyskiem-twardym/1351382
- https://docs.lavasoftware.org/lava/#architecture
- https://wiki.postgresql.org/wiki/File:PostgreSQL_logo.3colors.120x120.png
- https://www.djangoproject.com/m/img/logos/django-logo-negative.png
- https://www.servethehome.com/introducing-project-tinyminimicro-home-lab-revolution/