A Case for DAG Databases

FOSDEM 2023

Nikolai Kondrashov
Nikolai Kondrashov

_ssh ssh ssh ssh ssh ssh ssh ssh spbnick_

Sr. Software Engineer, Red Hat, CKI Project
KernelCI KCIDB developer & maintainer
Electronics and embedded enthusiast

Born in Russia, living in Finland
Kernel contribution workflow
Kernel testing systems
What are KernelCI and KCIDB
The Kernel's CI problem
Our solution
Our data model
Queries we need
Neo4j experience and ideas
Fallback plan
Kernel contribution workflow
Kernel contribution workflow

- Send patches by email to maillists and maintainers
- Get reviews in response
- Repeat until everyone is satisfied
- Get patches merged into a subtree
- Get subtree merged into the mainline
- *Maybe* get a report from a testing system at some point
Kernel testing systems
Kernel testing systems

LKFT

KernelCI

Intel 0-Day

Google syzbot

...
Kernel testing systems

stabil-rc/linux-4.9.1 build: 197 builds: 1 failed, 196 passed, 4 warnings (v4.9.232)

kernel.org bet [at] kernel.org
Sat Aug 18 08:43:50 UTC 2020

- Privacy message: [PATCH] stabilize/MDM: Logrotate Introductory and use [graphical for adhd]
- This message was generated by [patch-number]@ kernel.org 2019-06-24 16:52:46
- Messages sorted by: [Date] [Thread] [Order] [Subject]

stable-rc/linux-4.9.1 builds: 197 builds: 1 failed, 196 passed, 4 warnings (v4.9.232)


Re: [PATCH V4] hwmon: add fan/pwm driver for cors

KernelCI: kernel-usb-infoleak in hif_usb_send

PASS: Test report for kernel 5.7.12-3ff3d4f.cki (stable-queue)

Re: [PATCH V4] hwmon: add fan/pwm driver for cors

[Image]

KernelCI: kernel-usb-infoleak in hif_usb_send

PASS: Test report for kernel 5.7.12-3ff3d4f.cki (stable-queue)

Re: [PATCH V4] hwmon: add fan/pwm driver for cors

Kernel testing systems
What are KernelCI and KCIDB
What are KernelCI and KCIDB

Native tests from kernelci-core

LAVA labs
Static checks

KernelCI Backend

KCIDB
What are KernelCI and KCIDB

KernelCI

Submitter A
Submitter B
Submitter C
Submitter D

JSON
Checkouts
Builds
Tests
Issues
Incidents
Builds
Tests
Issues
Checkouts
Incidents

DB

Subscriptions

Dashboard
What are KernelCI and KCIDB

Data comes in any order
What are KernelCI and KCIDB

https://kcidb.kernelci.org/
What are KernelCI and KCIDB
What are KernelCI and KCIDB

https://kcidb.kernelci.org/
What are KernelCI and KCIDB

Below is the summary of results Kernel CI database has recorded for this revision so far. See complete and up-to-date report at:

https://kcidb.kernelci.org/d/revisions/revisions/80082538083206-39275af0a69370adb25ef1a556-patchset_hash=

OVERVIEW

Builds: X FAIL
Tests: X FAIL

REVISION

Status [FAIL]
Commits
name: v5.16.rc7-108-g90429388318
hash: 09082538083206-39275af0a69370adb25ef1a5
Checked out from

By
kernels, readat, syzbot, toxsuite

BUILDLS

Status
arches [X] 26 798
riscv 8 21
maps 7 11
powerpc 5 47
x86 1 11
arm 3 55
generic 12 96
aarch 1 37
ppc440 1 1
sh 1 24
sparc 3 12
sparc 12 32
arm64 2 3
x86_64 2

Failures
x vcc allocconfig
  1 are all/config v5.16.rc7-108-g90429388318
  1 x86 54 alloc/config 0.16.10-rc7-v9008253808318
  1 udg alloc/config v5.16.rc7-108-g90429388318
x pwrpc maple alloc/config
  2 pwrpc maple alloc/config

By
kernels, readat, syzbot, toxsuite
The Kernel's CI problem
The Kernel's CI problem

- Hardware is a natural scarcity
- Test results depend on hardware, timing, etc.
- A broken change might pass and a healthy change might fail
- Hard to remove noise from the results
The Kernel's CI problem

- Hard to investigate and reproduce
- Developers are skeptical of reports
- Reports undergo human reviews before sending
- Nobody stops development to fix CI
- Feedback loop is too long
The Kernel's CI problem

Ideal
The Kernel's CI problem
Our solution
Our solution
Our solution
Our solution
Our solution
Our solution

Kind: Kernel bug
Resolved: No
Policy: public (read: None, write: None)

Ticket URL: https://bugzilla.redhat.com/show_bug.cgi?id=2032094
Origin Tree: stable

Regexpes

<table>
<thead>
<tr>
<th>Text</th>
<th>File Name</th>
<th>Test Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[[s+FALL\s+]] :: Performance comparison: min:d+ * 1.15 &lt; max:d+</td>
<td>taskout.log</td>
<td>Storage block - storage f</td>
</tr>
</tbody>
</table>

Affected hosts

Affected Checkouts

<table>
<thead>
<tr>
<th>ID</th>
<th>Tree</th>
<th>Branch</th>
<th>Start Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>redhat:449002104</td>
<td>mainline.kern</td>
<td>master</td>
<td>Jan. 16, 2022, 8:26 a.m.</td>
</tr>
<tr>
<td>redhat:448605532</td>
<td>kernel-rt-rhe9</td>
<td></td>
<td>Jan. 15, 2022, 12:54 a.m.</td>
</tr>
<tr>
<td>redhat:448600281</td>
<td>rhe9</td>
<td></td>
<td>Jan. 15, 2022, 12:33 a.m.</td>
</tr>
</tbody>
</table>

Our solution

Dog tax!
Our data model
Our data model

Our data model

- Checkout
- Repo
- Branch
- Commit
- Patches
- Log
- ...

Our data model

Our data model

Our data model
Our data model
Our data model

Revision → Revision → Revision

Checkout → Checkout → Checkout

Build → Build → Build → Build → Build

Test → Test → Test → Test → Test

Test
Path
Environment
Status
Output files
Log
…

Our data model

Our data model

![Diagram showing the data model relationships between Revision, Checkout, Build, Test, Issue, Report, Culprit, Pattern, Status, and Version.](https://github.com/kernelci/kcidb-io/blob/main/kcidb_io/schema/v04_01.py)
Our data model

Our data model

Our data model

Our (planned) data model
Our (planned) data model
Our (planned) data model
Our (planned) data model
Our (planned) data model
Our (planned) data model

Commit parent
Implicit parent
Our (planned) data model

Commit parent
Implicit parent
Our (planned) data model

Commit parent
Implicit parent
Our (planned) data model

Directed Acyclic Graph (DAG)

Commit parent
Implicit parent
Our (planned) data model

- There's likely too many build and test results for a Graph Database.
- We might be able to keep issues there, but likely not incidents.
Queries we need
Queries we need

- Which issues should we triage for this revision?
  - We have to prioritize issues to triage.
- What were last X test results for last Y commits?
  - We gotta alert on changing result patterns.
  - Performance trends along revision history.
- Can we release this branch?
  - Which issues were found since last release?
- Which branch revisions have a particular issue?
Neo4j experience and ideas
Neo4j experience and ideas

```python
#!/usr/bin/env python3

import sys
import io
import csv

LOG = io.TextIOWrapper(sys.stdin.buffer, errors='replace')

def readfield(first=False):
    field = LOG.readline()
    if field == '':
        sys.exit((not first))
    return field[:-1]

with open(sys.argv[1], 'w') as commits_file,
     open(sys.argv[2], 'w') as relation_file:
    commits_writer = csv.writer(commits_file,
                                 delimiter=',', quotechar='"',
                                 escapechar='\', quoting=csv.QUOTE_MINIMAL)
    commits_writer.writerow(['hash',
                              'author_name', 'author_email', 'author_timestamp',
                              'subject'])
    relations_writer = csv.writer(relation_file,
                                   delimiter=',', quotechar='"',
                                   escapechar='\', quoting=csv.QUOTE_MINIMAL)
    relations_writer.writerow(['hash', 'parent_hash'])
    for line in LOG:
        line = line[:-1]
        fields = line.split()  # split on spaces
        commit_hash, author_name, author_email, author_timestamp,
        subject, parent_hashes = fields
        commits_writer.writerow([commit_hash,
                                  author_name, author_email, author_timestamp,
                                  subject])
        for parent_hash in parent_hashes.split():
            relations_writer.writerow([commit_hash, parent_hash])
```

Neo4j experience and ideas

```python
# /usr/bin/env python3

import sys
import io
import csv

LOG = io.TextIOWrapper(sys.stdout.buffer, encoding='replace')

def readfield(first=False):
    field = LOG.readline()
    if field == '':
        sys.exit(print('not first'))
    return field

with open(sys.argv[1], 'r') as commits_file, 
     open(sys.argv[2], 'r') as relations_file:
    commits_writer = csv.writer(commits_file, 
                                 doublequote=False, escapechar='\',
                                 quoting=csv.QUOTE_MINIMAL)
    relations_writer = csv.writer(relations_file, 
                                   doublequote=False, escapechar='\',
                                   quoting=csv.QUOTE_MINIMAL)

    commits_writer.writerow(['hash', 'author_name', 'author_email', 'author_timestamp', 'subject'])
    relations_writer.writerow(['hash', 'parent_hashes'])

    while True:
        commit_hash = readfield(True)
        parent_hashes = readfield()  
        author_name = readfield()  
        author_email = readfield()  
        author_timestamp = readfield()  
        subject = readfield()  
        commits_writer.writerow([commit_hash, author_name, author_email, author_timestamp, subject])

        for parent_hash in parent_hashes.split():
            relations_writer.writerow([commit_hash, parent_hash])

nkondras@ruby:~$ git log --format='%H%n%P%n%an%n%ae%n%at%n%s' | ./log2csv commits.csv relations.csv
```
Neo4j experience and ideas

```python
# !!/usr/bin/env python3
import sys
import io
import csv
LOG = io.TextIOWrapper(sys.stdout.buffer, errors='replace')

def readfield(first=False):
    field = LOG.readline()
    if field == '':
        sys.exit('not first')
    return field[:-1]

with open(sys.argv[1], '+') as commits_file,
     open(sys.argv[2], '+') as relations_file:
    commits_writer = csv.writer(commits_file,
                                  doublequote=False, escapechar='\',
                                  lineterminator='
')
    commits_writer.writerow(['hash', 'author_name', 'author_email', 'author_timestamp', 'subject'])
    relations_writer = csv.writer(relations_file,
                                   doublequote=False, escapechar='\',
                                   lineterminator='
')
    relations_writer.writerow(['hash', 'parent_hash'])

for commit in sys.stdin:
    commit_hash = readfield(True)
    parent_hashes = readfield()
    author_name = readfield()
    author_email = readfield()
    author_timestamp = readfield()
    subject = readfield()
    commits_writer.writerow([commit_hash, author_name, author_email, author_timestamp, subject])
    for parent_hash in parent_hashes.split():
        relations_writer.writerow([commit_hash, parent_hash])
```


nkondras@ruby:linux$ git log --format=\"%H%n%P%n%an%n%ae%n%at%n%s\" | ./.log2csv commits.csv
nkondras@ruby:linux$ wc -l commits.csv
1154921 commits.csv
Neo4j experience and ideas

```python
#!/usr/bin/env python3

import sys
import io
import csv

LOG = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')

def readfield(default=False):
    field = LOG.readline()
    if field == '':
        sys.exit('exit(0)')
    return field

with open(sys.argv[1], 'r') as commits_file:
    commits = csv.writer(sys.stdout, dialect='excel', escapechar='\')
    commits.writerow(['hash', 'author_name', 'author_email', 'author_timestamp',
                      'commit', 'commit_message'])
    for commit in commits_file:
        hash, author_name, author_email, author_timestamp, commit_message = commit.split()""
        commits.writerow([hash, author_name, author_email, author_timestamp, commit_message])
```

**Neo4j experience and ideas**

```python
#!/usr/bin/env python

import sys
import io
import csv

LOG = io.TextIOWrapper(sys.stdout.buffer, errors='replace')

def readfield(first=False):
    field = LOG.readline()
    if field == '':
        sys.exit((not first))
    return field[1:]

with open(sys.argv[1], 'r') as commits_file,
     open(sys.argv[2], 'w') as relations_file:
    commits_writer = csv.writer(commits_file, quoting=csv.QUOTE_ALL)
    relations_writer = csv.writer(relations_file, quoting=csv.QUOTE_ALL)

    commits_writer.writerow(['hash', 'author_name', 'author_email', 'author_timestamp', 'subject'])
    relations_writer.writerow(['hash', 'parent_hash'])

    while True:
        commit_hash = readfield(True)
        author_name = readfield()
        author_email = readfield()
        author_timestamp = readfield()
        subject = readfield()
        commits_writer.writerow([commit_hash, author_name, author_email, author_timestamp, subject])

        for parent_hash in parent_hashes.split():
            relations_writer.writerow([commit_hash, parent_hash])
```

nkondras@ruby:~$ git log --format='%%h%n%%an%n%%ae%n%%at' | ./log2csv

nkondras@ruby:~$ wc -l commits.csv
154921 commits.csv

nkondras@ruby:~$ head commits.csv
hash,author_name,author_email,author_timestamp,subject
```

nkondras@ruby:~$ wc -l relations.csv
1247839 relations.csv
```

Neo4j experience and ideas


nkondras@ruby:~/linux$ git log --format='%H%n%P%n%an%n%ae%n%an%n%ks' | ./log2csv commits.csv relations.csv
nkondras@ruby:~/linux$ wc -l commits.csv
1154921 commits.csv

nkondras@ruby:~/linux$ head commits.csv
hash, author_name, author_email, author_timestamp, subject
00675345c95c3777a50b0a00b9e5e85d72e2b7d, Linus Torvalds, torvalds@linux-foundation.org, 2023-01-31T16:02:05-08:00, Merge tag 'c e5aee808034bf7b8bfe9d744a417a4b27b7fe222a, Haiman Long, longman@redhat.com, 2023-01-31T18:48:03-05:00, cgroup/cpuset: Fix wrong c
58796f7f04b57019bada81af17f372189315f6e5, Linus Torvalds, torvalds@linux-foundation.org, 2023-01-31T08:00-08:00, Merge tag 'b 88b35e61e1919dca70b01180ad0b83847b0d64a, Linus Torvalds, torvalds@linux-foundation.org, 2023-01-31T17:38-08:00, Merge tag 'm
22b07b7d0f8e08c65ed008f08ef7e7e5a4c2ad5a, Linus Torvalds, torvalds@linux-foundation.org, 2023-01-30T11:58-53-08:00, Merge tag 'f 328866c7258fe1c7f5c6e33df3ab18009c8e7, Hou Tao, houtao@huawei.com, 2023-01-31T19:52:11-08:00, fscache: Use clear_and_wake_up
2826e37d2f43657da34d377e2b38f2024ada7, Hou Tao, houtao@huawei.com, 2023-01-31T19:52:10-08:00, fscache: Use wait_on_bit() to 6d79c50f84ca7f97122b11799e5a5710c4700, Linus Torvalds, torvalds@linux-foundation.org, 2023-01-29T13:59-43-08:00, Linux 6.2-rc ab8726a1eabfece8a932d4ba1a1a72ad2a6d4ca4, Linus Torvalds, torvalds@linux-foundation.org, 2023-01-29T11:26:49-08:00, Merge tag 'i
nkondras@ruby:~/linux$ wc -l relations.csv
1247839 relations.csv

nkondras@ruby:~/linux$ head relations.csv
hash, parent_hash
00675345c95c3777a50b0a00b9e5e85d72e2b7d, 00675345c95c3777a50b0a00b9e5e85d72e2b7d
58796f7f04b57019bada81af17f372189315f6e5, 00675345c95c3777a50b0a00b9e5e85d72e2b7d
e5aee808034bf7b8bfe9d744a417a4b27b7fe222a, 0808034bf7b8bfe9d744a417a4b27b7fe222a
58796f7f04b57019bada81af17f372189315f6e5, 88b35e61e1919dca70b01180ad0b83847b0d64a
88b35e61e1919dca70b01180ad0b83847b0d64a, 22b07b7d0f8e08c65ed008f08ef7e7e5a4c2ad5a
22b07b7d0f8e08c65ed008f08ef7e7e5a4c2ad5a, 09c000ee8b0dc5f9c8ad8d7ca92c9db1f20b0
22b07b7d0f8e08c65ed008f08ef7e7e5a4c2ad5a, 6d79c50f84ca7f97122b11799e5a5710c4700
22b07b7d0f8e08c65ed008f08ef7e7e5a4c2ad5a, 328866c7258fe1c7f5c6e33df3ab18009c8e7
nkondras@ruby:~/linux$
Neo4j experience and ideas

```cypher
// Load commits
:cypher: auto LOAD CSV WITH HEADERS FROM 'file:///commits.csv' AS commit
CALL {
  WITH commit
  CREATE (c:Commit)
  SET c = commit
}
IN TRANSACTIONS
```

Added 1154920 labels, created 1154920 nodes, set 5774600 properties, completed after 46692 ms.
Neo4j experience and ideas

```bash
mainline$ CREATE TEXT INDEX FOR (c:Commit) ON [c.hash]
```

Added 1 index, completed after 24 ms.
Load relations

:cypher

LOAD CSV WITH HEADERS FROM 'file:///relations.csv' AS relation

CALL {

WITH relation

MATCH (child:Commit {hash:relation.hash})

MATCH (parent:Commit {hash:relation.parent_hash})

CREATE (child)-[:PARENT]->(parent)

} IN TRANSACTIONS
Neo4j experience and ideas
Neo4j experience and ideas

MATCH (s:Commit {hash: '3123109284176b1532874591f7c81f3837bbdc17'})
MATCH (e:Commit {hash: 'd0d96121d03d6d9cf608d948247a9f24f5a02da9'})
WHERE exists((s)-[:PARENT*0 ..]→(e))
RETURN true
MATCH (s:Commit {hash:'3123109284176b1532874591f7c81f3837bbdc17'})
MATCH (e:Commit {hash:'68e77ffbfd06ae3ef8f2abf1c3b971383c866983'})
CALL apoc.path.expandConfig(s, {
  relationshipFilter: 'PARENT>',
  uniqueness: 'NODE_GLOBAL',
  terminatorNodes: [e],
  limit: 1
})
YIELD path
RETURN length(path)
MATCH (s:Commit {hash:'68e77ffbfd06ae3ef8f2abf1c3b971383c866983'})
MATCH (e:Commit {hash:'1e20904e417738066b26490de2daf7ef3ed34483'})
CALL apoc.path.expandConfig(s, {
  relationshipFilter: 'PARENT>',
  uniqueness: 'NONE',
  terminatorNodes: [e]
})
YIELD path
UNWIND nodes(path) AS node
RETURN count(DISTINCT node)
Neo4j experience and ideas

nkondras@ruby:linux$ time git log --oneline 1e20904e417738066b26490de2daf7ef3ed34483..68e77ffbfd06ae3ef8f2abf1c3b971383c866983 | wc -l
real 0m0.302s
user 0m0.268s
sys 0m0.038s
nkondras@ruby:linux$
Neo4j experience and ideas

● Add support for DAG-restricted databases
  ○ It is also possible to check whether a given directed graph is a DAG in linear time, either by attempting to find a topological ordering and then testing for each edge whether the resulting ordering is valid\(^{[18]}\) or alternatively, for some topological sorting algorithms, by verifying that the algorithm successfully orders all the vertices without meeting an error condition.\(^{[17]}\)

● Gradually add optimizations
  ○ Topological sorting is the algorithmic problem of finding a topological ordering of a given DAG. It can be solved in linear time.\(^{[16]}\)

  – Wikipedia
Fallback plan
Fallback plan

- Store all revisions as commits in a giant git repo
  - Branches
    - And their force-pushes
  - Patchsets
    - And their versions
- Query with libgit2
- Shuttle commit hashes to/from the relational DB
Thank you!
Join Us!
Main repos on GitHub

https://github.com/kernelci/?q=kcidb
A bunch of "good first issues"
#kernelci at libera.chat

IRC channel