Widespread Error Detection in Large Scale Continuous Integration Systems

- Stanislaw Swierc (stansw)
- James Lu (luchangj)
- Thomas Yi (thomasyi)

* @meta.com

https://github.com/StanislawSwierc/CCIW2024-Widespread-Error-Detection

Disclaimer

> *This presentation contains sample code with a basic pipeline for detecting widespread errors in the React open-source project.*
>
> *It does NOT contain any code, logs or data internal to Meta Platforms, Inc.*

Motivation

> *Improve developer productivity by quickly detecting and mitigating the impact of widespread errors.*

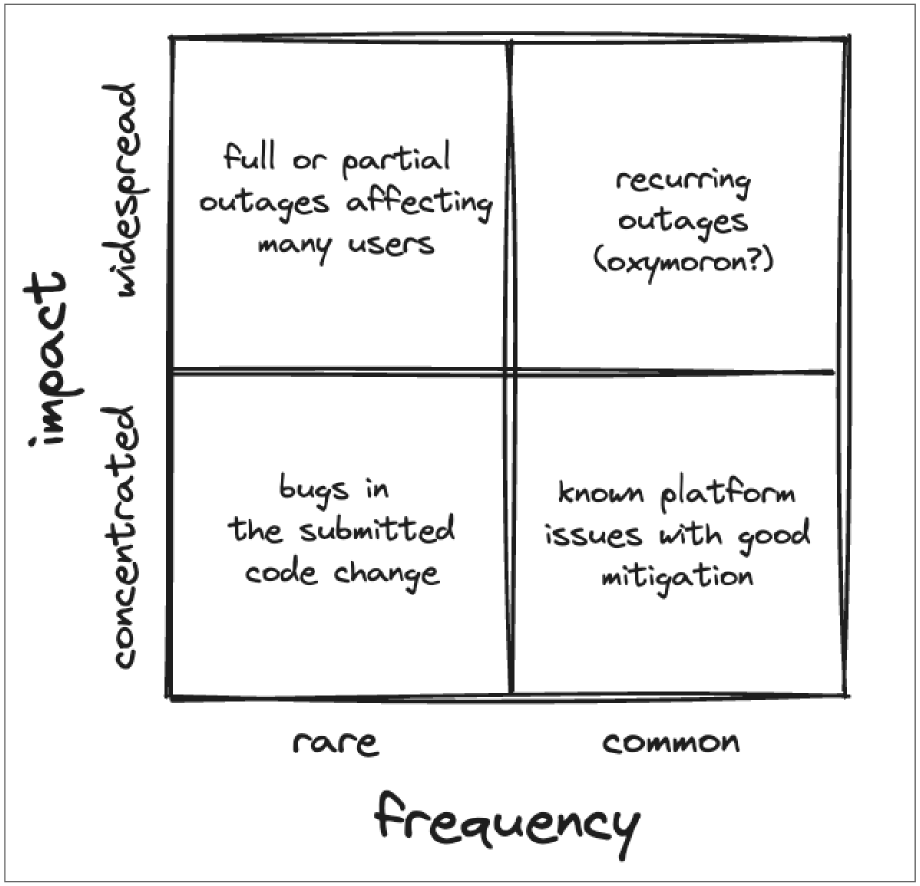
Motivation: Widespread Errors

Widespread errors are commonly infrastructural errors that falsely block developers from integrating their code changes into the mainline branch.
Why is this bad?

- **Productivity** - Developers waste time trying to fix code, when it is not at fault.
- **Trust** - Erodes confidence in CI System's ability to produce validation signals.
- **Efficiency** - Need to re-run failing validation, thus, wasting machine resources.

Motivation: Impact vs. Frequency

We can organize errors by number of affected people and how common they are:



	rare	common
widespread	full or partial outages affecting many users	recurring outages (oxymoron?)
concentrated	bugs in the submitted code change	known platform issues with good mitigation

Challenges

- Rare, widespread errors may appear like a group of concentrated errors.

- Capturing signatures of rare, widespread errors is only good for retroactive analysis, but not for detection.

React Case Study

React uses CircleCI to validate pull requests and the health of the mainline branch.

- **Project** - validation configuration for the code repository
    - **Pipeline** - group of workflows
        - **Workflow** - graph of jobs
            - **Job** - set of steps executed sequentially
                - **Step/Action** - execution unit
                    - **Logs** - standard output and summary

https://circleci.com/docs/concepts

React Case Study: CircleCI API

CircleCI offers an API to fetch information about pipelines and other resources all the way to individual logs.

In [4]:

```
from pycircleci.api import Api

ORG = "facebook"
PROJECT = "react"
TOKEN = "{token}"

client = Api(token=TOKEN)
```

In [5]:

```
%%script true

client.get_project_pipelines(...
client.get_pipeline_workflow(...
client.get_workflow_jobs(...
client.get_job_details_v1(...
client._session.get(action["output_url"])
```

In [12]:

```
!du -h pipelines.jsonl
```

```
4.4G    pipelines.jsonl
```

https://circleci.com/docs/api/v2/index.html

https://circleci.com/docs/managing-api-tokens

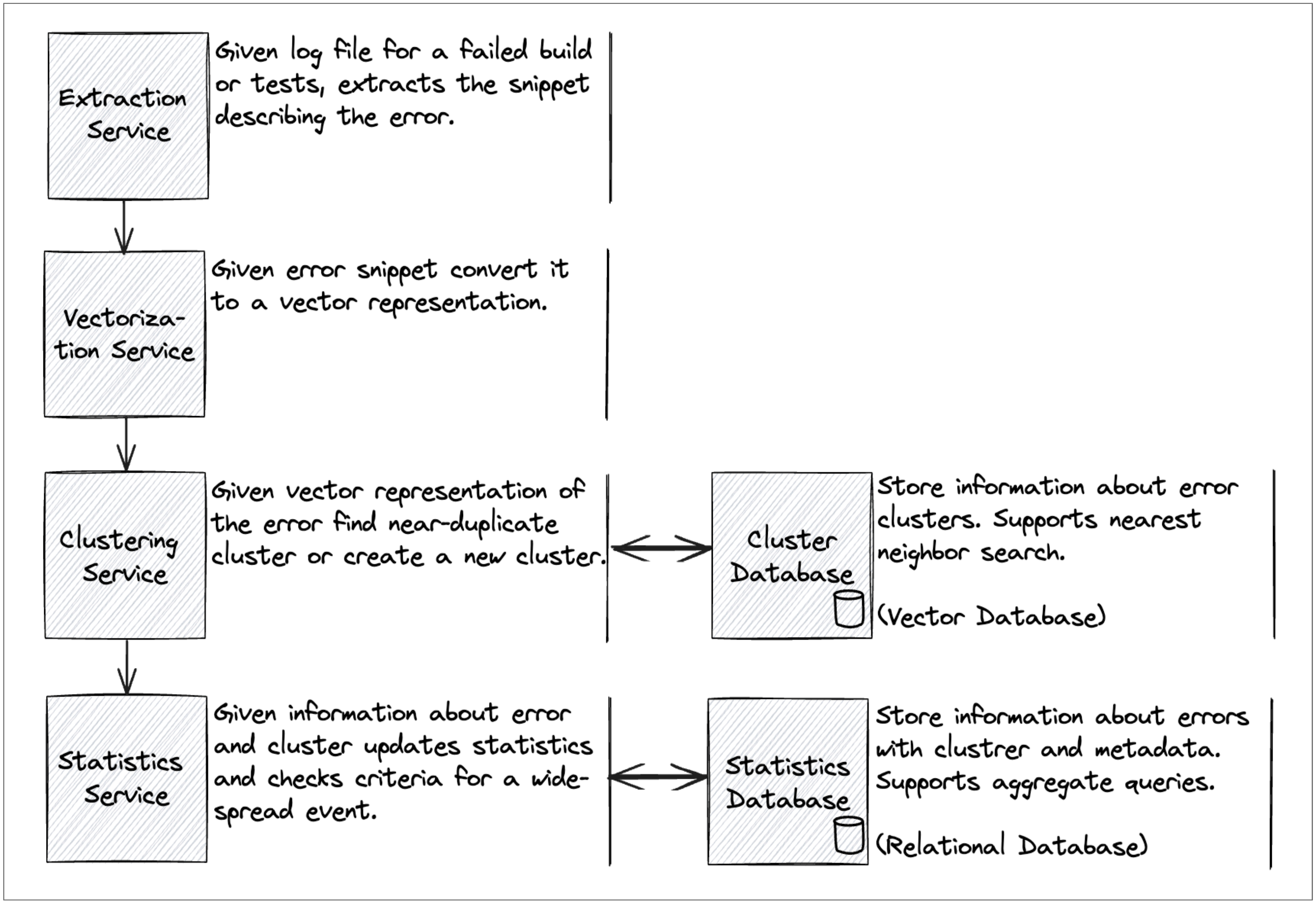
React Case Study: Actions Fact Table

In [13]:

```
actions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 341630 entries, 0 to 341629
Data columns (total 17 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   pipeline_id                   341630 non-null  object
 1   pipeline_actor_login          341630 non-null  object
 2   workflow_id                   341630 non-null  object
 3   workflow_name                 341630 non-null  object
 4   workflow_status               341630 non-null  object
 5   job_id                        341630 non-null  object
 6   job_name                      341630 non-null  object
 7   job_status                    341630 non-null  object
 8   job_link                      341630 non-null  object
 9   action_name                   341630 non-null  object
 10  action_fully_qualified_name   341630 non-null  object
 11  action_index                  341630 non-null  int64
 12  action_status                 341630 non-null  object
 13  action_output_time            341630 non-null  datetime64[ns, UTC]
 14  action_output_message         341630 non-null  object
 15  action_output_type            341630 non-null  object
 16  action_output_date            341630 non-null  datetime64[ns, UTC]
dtypes: datetime64[ns, UTC](2), int64(1), object(14)
memory usage: 44.3+ MB
```

Pipeline Design



Extraction Service
Given log file for a failed build or tests, extracts the snippet describing the error.
Vectorization Service
Given error snippet convert it to a vector representation.
Clustering Service
Given vector representation of the error find near-duplicate cluster or create a new cluster.
Cluster Database
Store information about error clusters. Supports nearest neighbor search.
(Vector Database)
Statistics Service
Given information about error and cluster updates statistics and checks criteria for a wide-spread event.
Statistics Database
Store information about errors with clustrer and metadata. Supports aggregate queries.
(Relational Database)

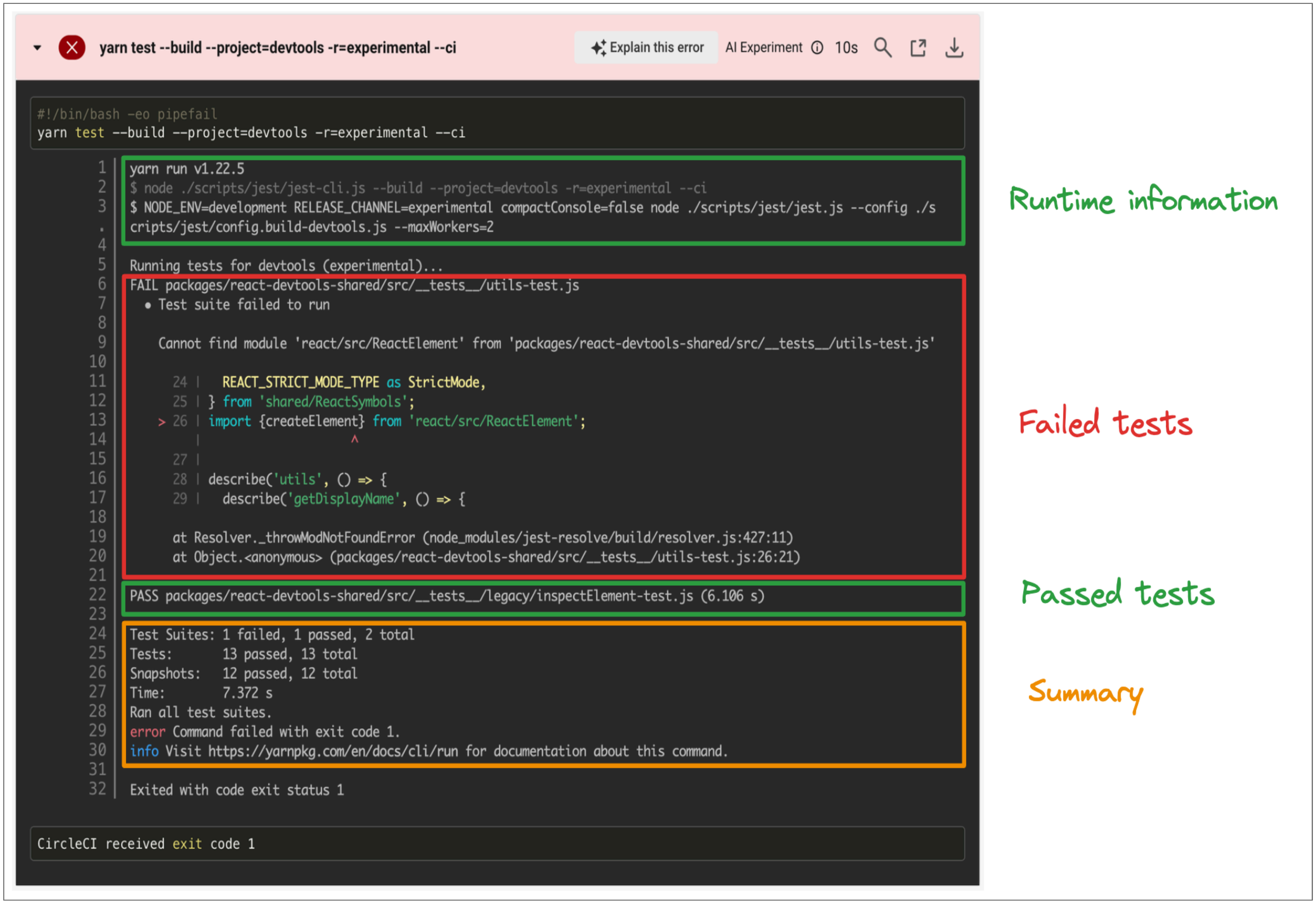
Extraction

Extraction

Given a log file for a failed build or tests, extracts the snippet describing the error.

1. **Federated solution** - allow teams to provide custom rules for extracting relevant error snippets.
2. **General solution** - find lines which appear only in logs of failed actions.

Extraction: Example



yarn test --build --project=devtools -r=experimental --ci

```
#!/bin/bash -eo pipefail
yarn test --build --project=devtools -r=experimental --ci
```

```
yarn run v1.22.5
$ node ./scripts/jest/jest-cli.js --build --project=devtools -r=experimental --ci
$ NODE_ENV=development RELEASE_CHANNEL=experimental compactConsole=false node ./scripts/jest/jest.js --config ./s
cripts/jest/config.build-devtools.js --maxWorkers=2

Running tests for devtools (experimental)...
FAIL packages/react-devtools-shared/src/__tests__/utils-test.js
  • Test suite failed to run

    Cannot find module 'react/src/ReactElement' from 'packages/react-devtools-shared/src/__tests__/utils-test.js'

      24 |   REACT_STRICT_MODE_TYPE as StrictMode,
      25 | } from 'shared/ReactSymbols';
    > 26 | import {createElement} from 'react/src/ReactElement';
         |                             ^
      27 |
      28 | describe('utils', () => {
      29 |   describe('getDisplayName', () => {

      at Resolver._throwModNotFoundError (node_modules/jest-resolve/build/resolver.js:427:11)
      at Object.<anonymous> (packages/react-devtools-shared/src/__tests__/utils-test.js:26:21)

PASS packages/react-devtools-shared/src/__tests__/legacy/inspectElement-test.js (6.106 s)

Test Suites: 1 failed, 1 passed, 2 total
Tests:       13 passed, 13 total
Snapshots:   12 passed, 12 total
Time:        7.372 s
Ran all test suites.
error Command failed with exit code 1.
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.

Exited with code exit status 1
```

```
CircleCI received exit code 1
```

Runtime information

Failed tests

Passed tests

Summary

Extraction: Sample Code

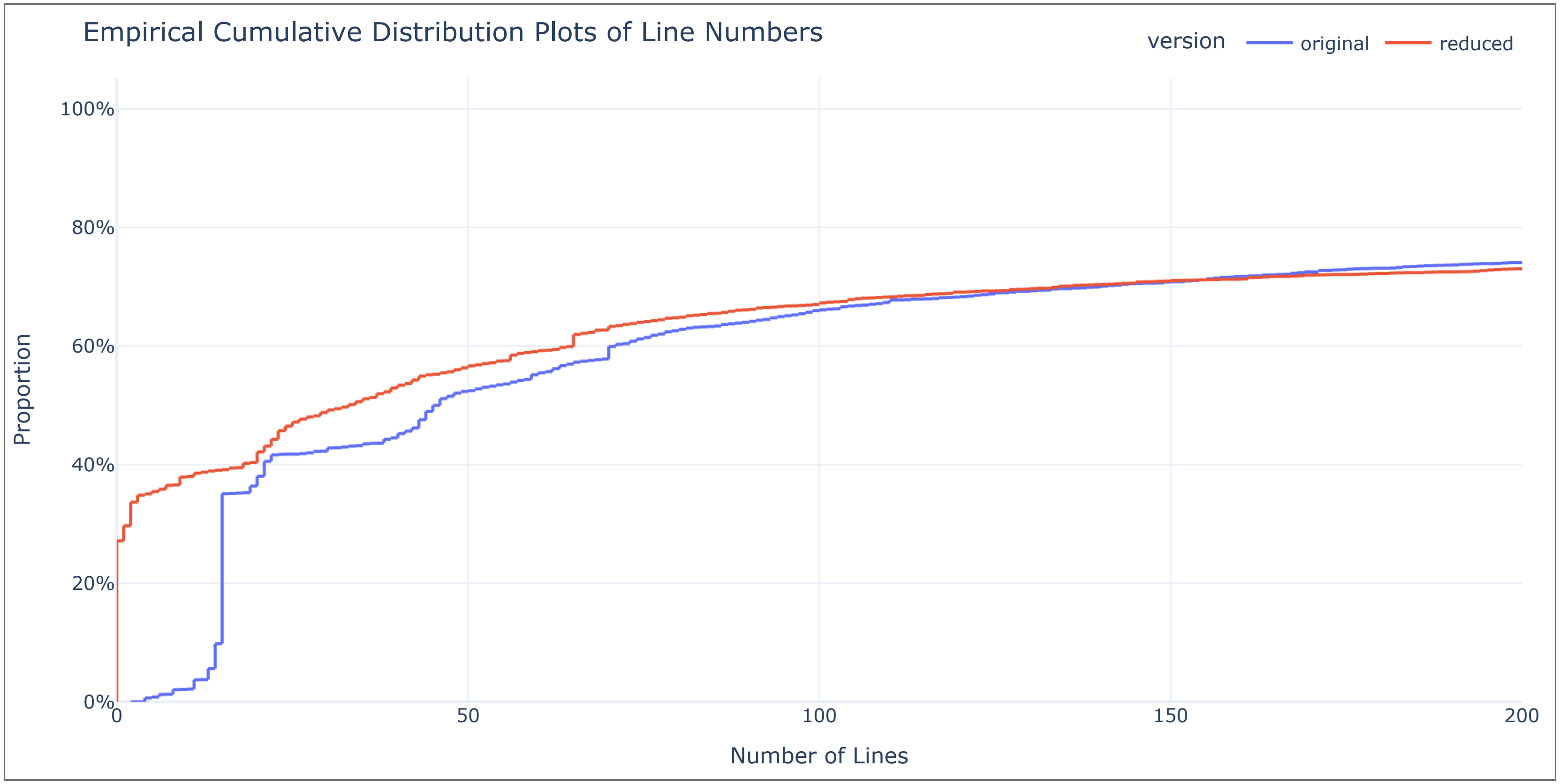
In [15]:

```python
def assign_action_output_failure(df, cache_size=2048):
    result = []
    action_cache = defaultdict(lambda: cachetools.LRUCache(cache_size))
    for i, row in tqdm(df.iterrows(), total=len(df)):
        failure_lines = []
        success_cache = action_cache[row["action_fully_qualified_name"]]
        for line in row["action_output_message"].splitlines():
            template = parse_line(line)

            if row["action_status"] == "success":
                success_cache[template] = True
            elif not template or template not in success_cache:
                failure_lines.append(line)

        result.append("\n".join(failure_lines))
    return result
```

Extraction: Results

In [69]:

```
line_num_ecdf_fig.show()
```



Extraction: Extensions

1. Use log parser and diff failures based on log templates

   Jiang, Z., Liu, J., Huang, J., Li, Y., Huo, Y., Gu, J., Chen, Z., Zhu, J. and Lyu, M.R., 2023.

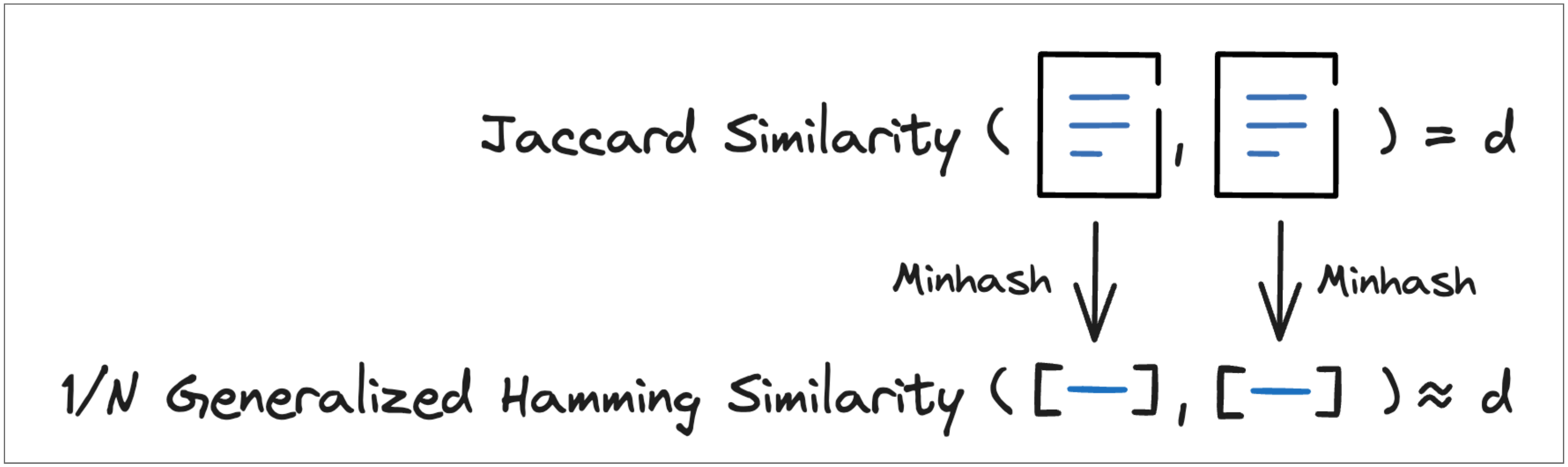
   A Large-scale Benchmark for Log Parsing

Vectorization

Vectorization

Convert raw text data into a numerical format that can be efficiently processed.

1. Bag-of-word model with an open-vocabulary
2. Minhash (e.g. with 256 x 16-bit we get a constant 0.5 kB per error)


Jaccard Similarity (,) = d
Minhash
Minhash
1/N Generalized Hamming Similarity ([—], [—]) ≈ d

Leskovec J, Rajaraman A, Ullman JD. "Chapter 3: Finding Similar Items" Mining of massive data sets. Cambridge university press; 2020 Jan 9.

Vectorization: Normalization

Reduce the size of the vocabulary by removing common hashes and common metadata.

In [20]:

```
normalize_rules = [
    (
        "{GUID}",
        re2.compile(r'[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}')
    ),
]

def normalize(text):
    for repl, regex in normalize_rules:
        text = regex.sub(repl, text)
    return text
```

Vectorization: Tokenization

Extract meaningful tokens.

In [21]:

```
token_regex = re2.compile(r"\b[[:alpha:]_][[:alpha:][:digit:]_-]+")

def tokenize(text):
    normalized_text = normalize(text)
    return token_regex.findall(normalized_text)
```

Vectorization: Minhash

Convert set of tokens to Minhash sketches.

In [22]:

```
from datasketch import MinHash

NUM_PERM = 256

def vectorize(tokens):
    m = MinHash(num_perm=NUM_PERM)
    for token in set(tokens):
        m.update(token.encode('utf8'))
    return m.hashvalues
```

Vectorization: Results

Minhash representation is more compact and allows for fast nearest-neighbors search.
Total size of original text data:

In [23]:

```
print("{:.3f} GB".format(actions["action_output_message"].str.len().sum() / 2 ** 30))
```

```
1.978 GB
```

Total size of minhash sketches:

In [24]:

```
print("{:.3f} GB".format(actions.shape[0] * np.uint16(0).nbytes * NUM_PERM / 2 ** 30))
```

```
0.163 GB
```

Vectorization: Extensions

1. **Process documents in parallel.** Vectorization is embarrassingly parallelizable.
2. **Use Weighted Minhash**. Minhash with Inverse Document Frequency (IDF) weights can automatically ignore common terms such as "failed".
   Chum, O., Philbin, J. and Zisserman, A., 2008, September. Near duplicate image detection: Min-hash and TF-IDF weighting.
3. **Use Neural Network Vector Embeddings (future).** Embeddings are a good alternative to Minhash sketches and they can capture semantics of logs.
   Meng, W., Liu, Y., Huang, Y., Zhang, S., Zaiter, F., Chen, B. and Pei, D., 2020, August. A semantic-aware representation framework for online log analysis.

Clustering

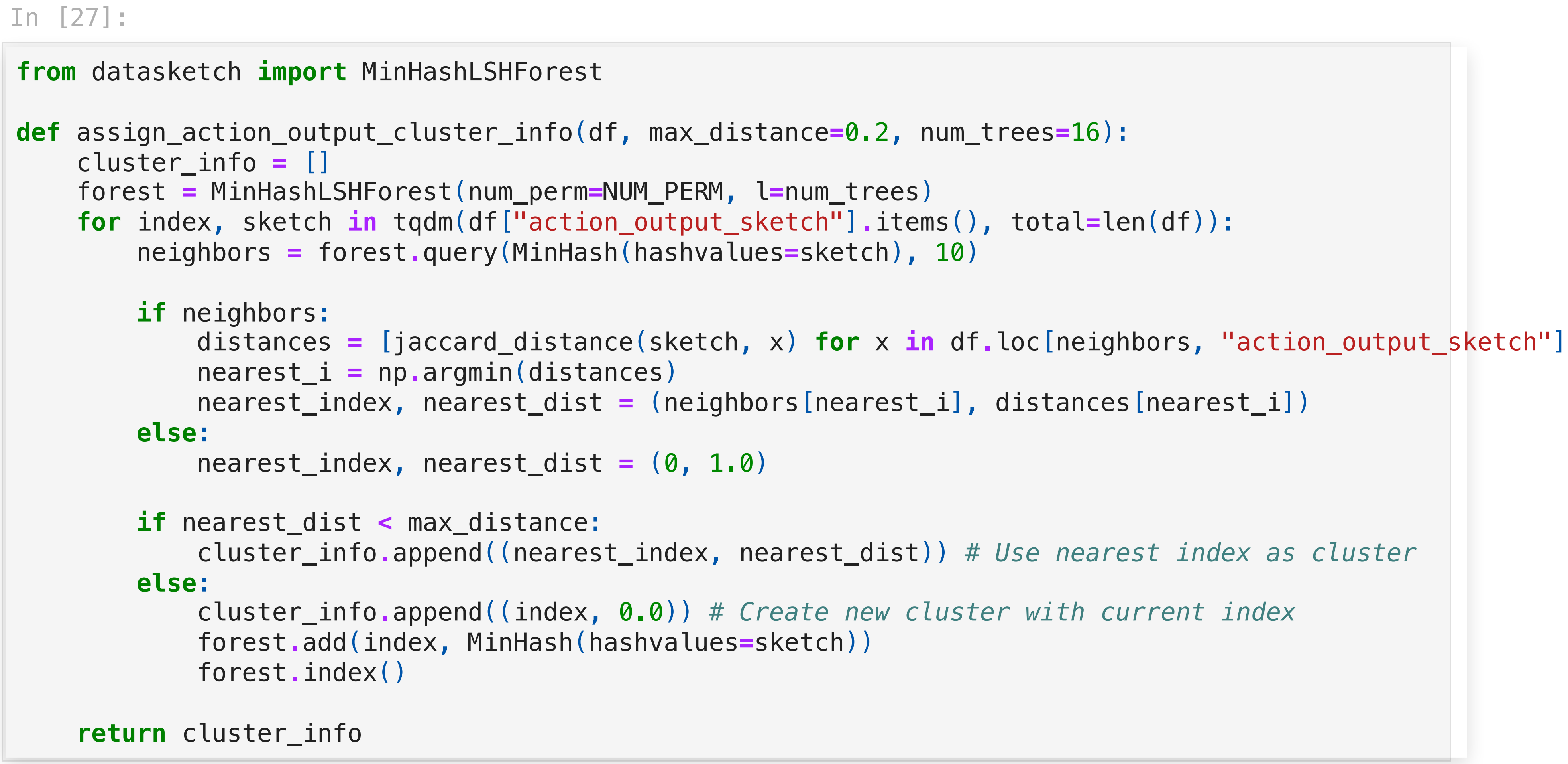
Clustering

Given vector representation of the error, find a **near-duplicate cluster** or create a new cluster.

- Near-duplicate search is performed on a vector database.
- For Minhash the database should support *Generalized Hamming Distance*.
- LSH Forest is an efficient data structure and offers time complexity of $O(n\log(n))$.

Bawa, M., Condie, T. and Ganesan, P., 2005, May. LSH forest: self-tuning indexes for similarity search.

Clustering: Sample Code



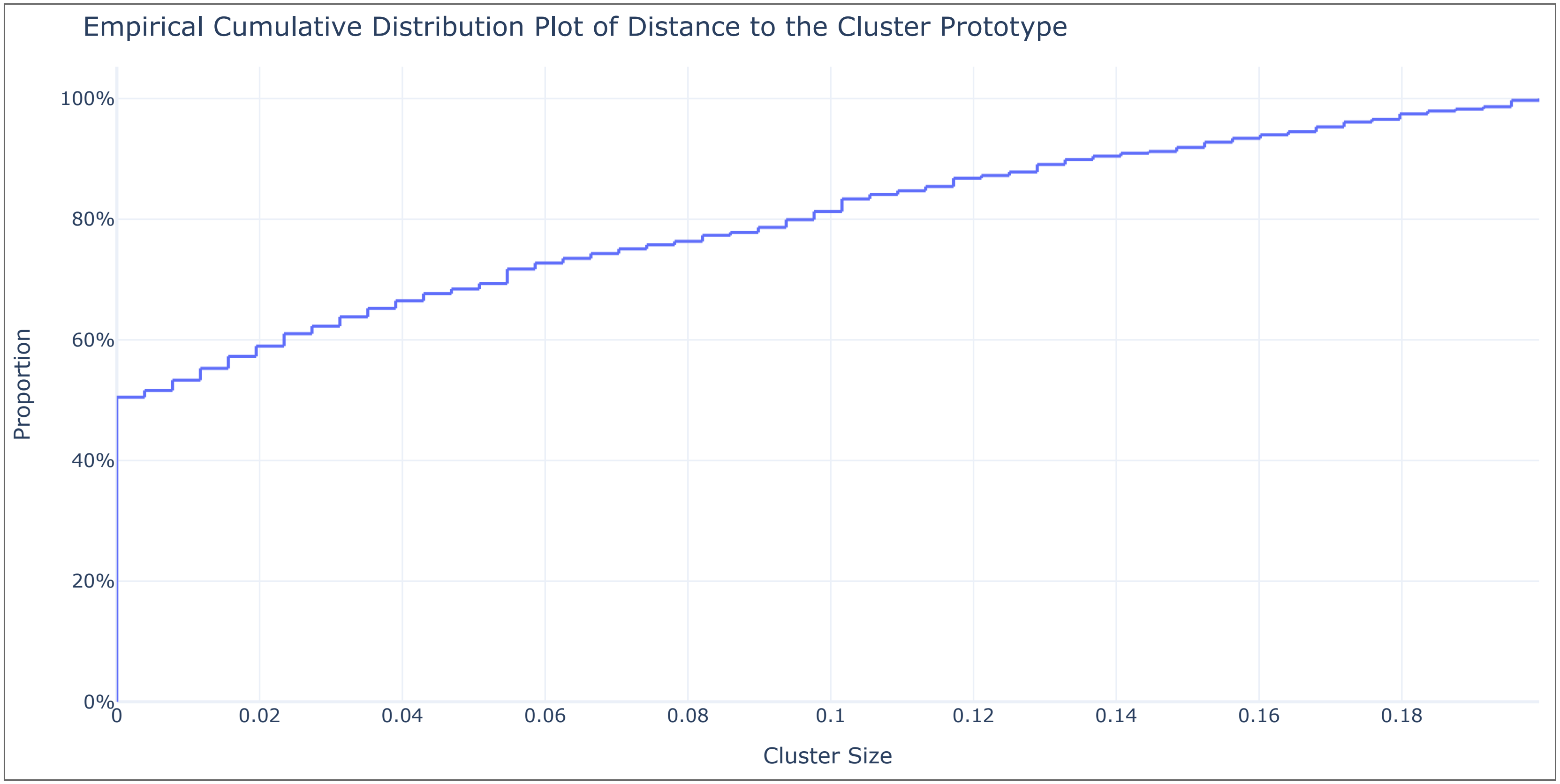
In [27]:

```
from datasketch import MinHashLSHForest

def assign_action_output_cluster_info(df, max_distance=0.2, num_trees=16):
    cluster_info = []
    forest = MinHashLSHForest(num_perm=NUM_PERM, l=num_trees)
    for index, sketch in tqdm(df["action_output_sketch"].items(), total=len(df)):
        neighbors = forest.query(MinHash(hashvalues=sketch), 10)

        if neighbors:
            distances = [jaccard_distance(sketch, x) for x in df.loc[neighbors, "action_output_sketch"]
            nearest_i = np.argmin(distances)
            nearest_index, nearest_dist = (neighbors[nearest_i], distances[nearest_i])
        else:
            nearest_index, nearest_dist = (0, 1.0)

        if nearest_dist < max_distance:
            cluster_info.append((nearest_index, nearest_dist)) # Use nearest index as cluster
        else:
            cluster_info.append((index, 0.0)) # Create new cluster with current index
            forest.add(index, MinHash(hashvalues=sketch))
            forest.index()

    return cluster_info
```

Clustering: Results

In [62]:

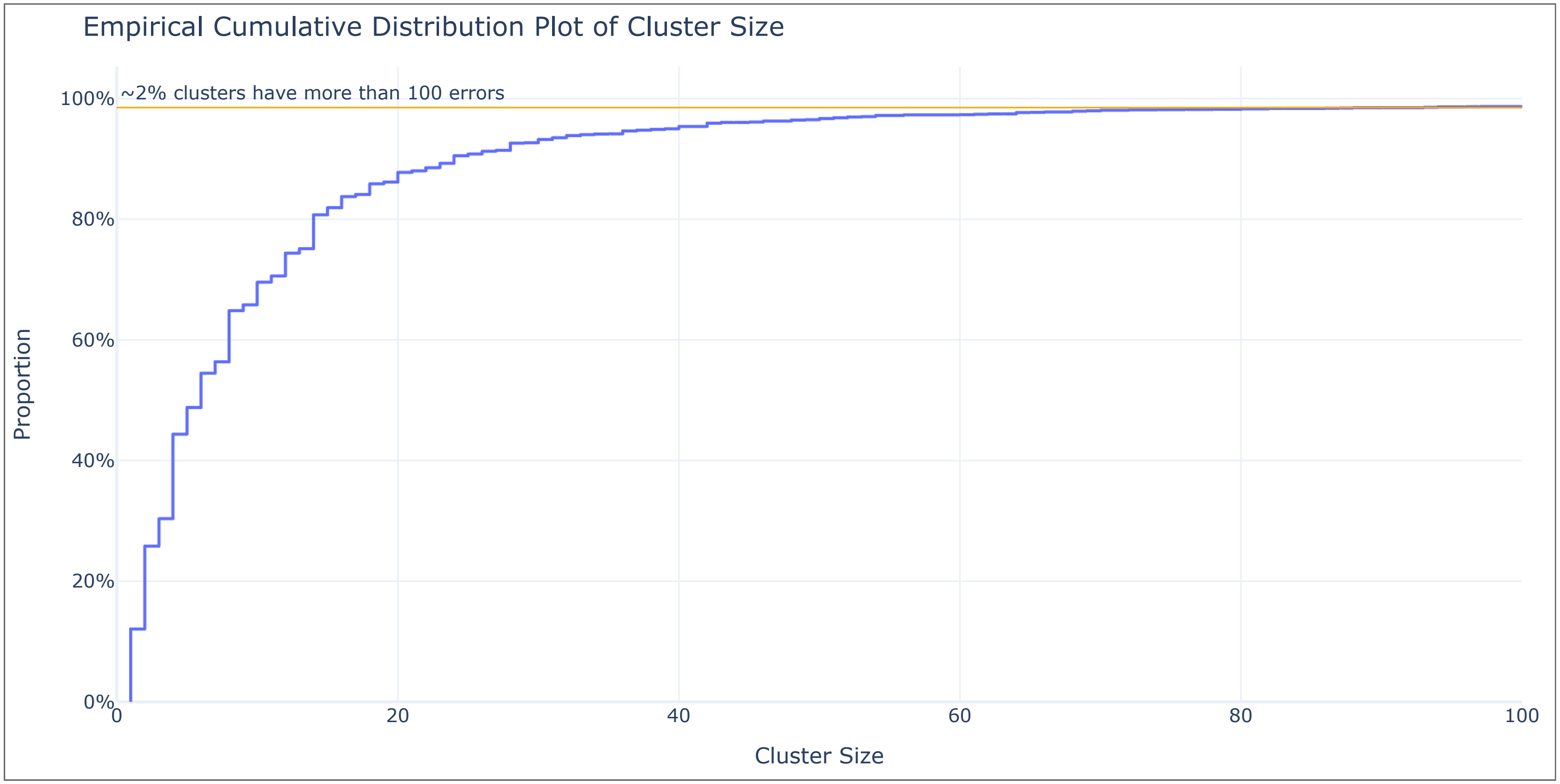
```
nearest_distance_ecdf_fig.show()
```



Clustering: Results

In [66]:

```
cluster_size_ecdf_fig.show()
```



Clustering: Extensions

1. **Partition clusters by creation time.** Clustering is an append only process. This opens up the opportunity to partition clusters by creation time to optimize database access.
2. **Detect and fix duplicate clusters.** If many similar errors get processed at the same time, they may end up creating many clusters. There is a need of a background job which will detect and fix duplicate clusters.
3. **Use different distance thresholds for different classes of errors.** Different frameworks may produce different logs and they may benefit from using criteria for near-duplicate failure.
4. **Use Least Recently Used (LRU) policy to manage cluster lifetime**. Most clusters represent unique errors and can be safely deleted after just a few days.

Statistics

Statistics: Top clusters by failed actions

In [34]:

```
cluster_stats.head(10)
```

Out[34]:

	action_output_cluster_id	unique_users_num	actions_num	distance_avg
0	8	21	15966	0.000000
1	316553	9	1948	0.056021
2	176294	2	1441	0.052001
3	306798	6	1329	0.064040
4	87844	1	1140	0.000021
5	129316	2	985	0.067536
6	131692	1	775	0.109340
7	322598	1	765	0.140508
8	313562	5	732	0.017306
9	333598	1	432	0.056993

The largest cluster comprises of failures with no message:

In [35]:

```
empty_cluster_id = 8
```

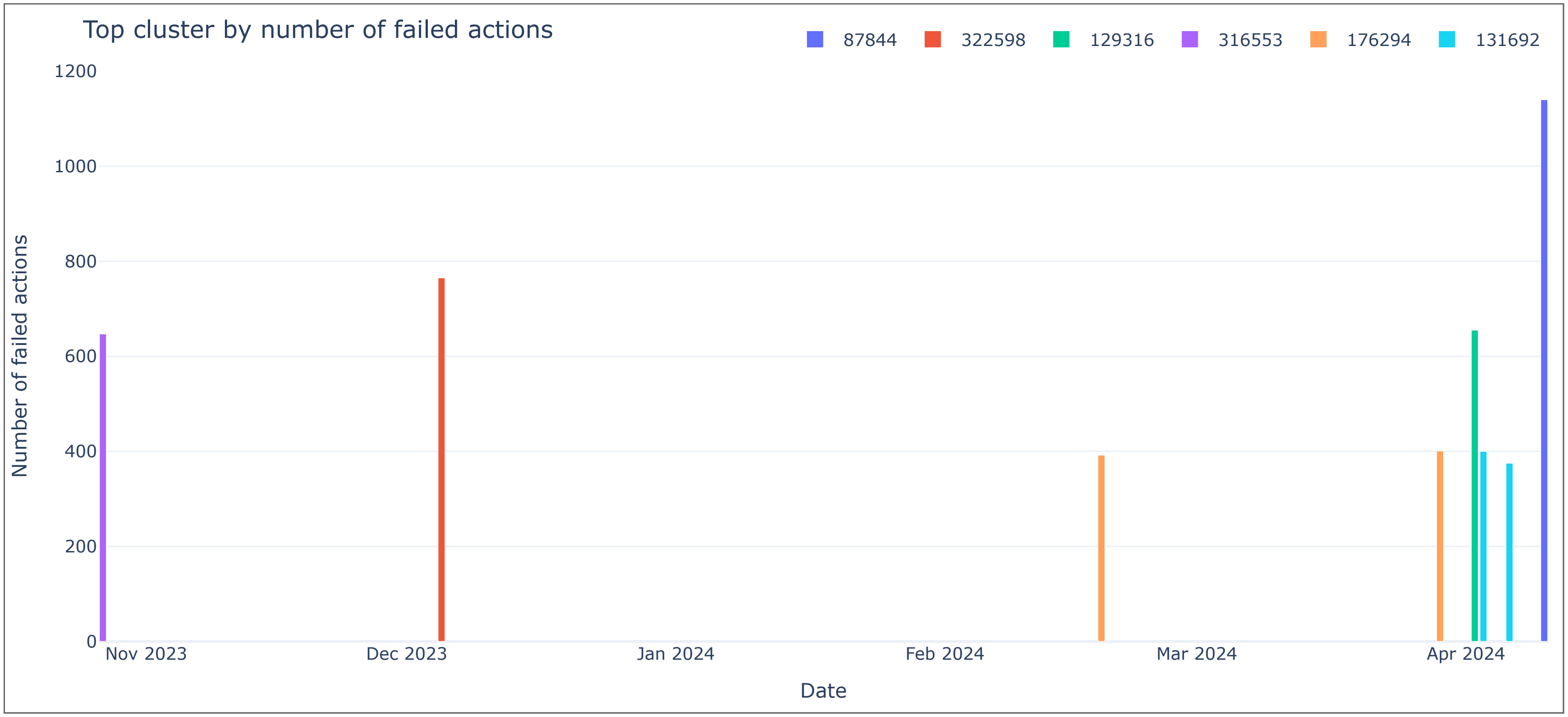
```
print(repr(actions.loc[empty_cluster_id, "action_output_failure"]))
```

```
''
```

Statistics: Top clusters by failed actions

In [38]:

```
top_clusters_by_number_of_actions_fig.show()
```



Statistics: Top clusters by failed actions

In [40]:

```
print(actions.loc[129316, "action_output_failure"])
```

```
error eslint-v9@9.0.0: The engine "node" is incompatible with this module. Expected
version "^18.18.0 || ^20.9.0 || >=21.1.0". Got "16.16.0"
error Found incompatible module.
info Visit https://yarnpkg.com/en/docs/cli/install for documentation about this comm
and.
```

In [41]:

```
print("\n".join(actions.loc[176294, "action_output_failure"].splitlines()[:7]))
```

```
FAIL packages/react-reconciler/src/__tests__/ReactIncrementalUpdates-test.js
  ● Test suite failed to run

    Jest encountered an unexpected token

    Jest failed to parse a file. This happens e.g. when your code or its dependencie
s use non-standard JavaScript syntax, or when Jest is not configured to support such
syntax.
```

In [42]:

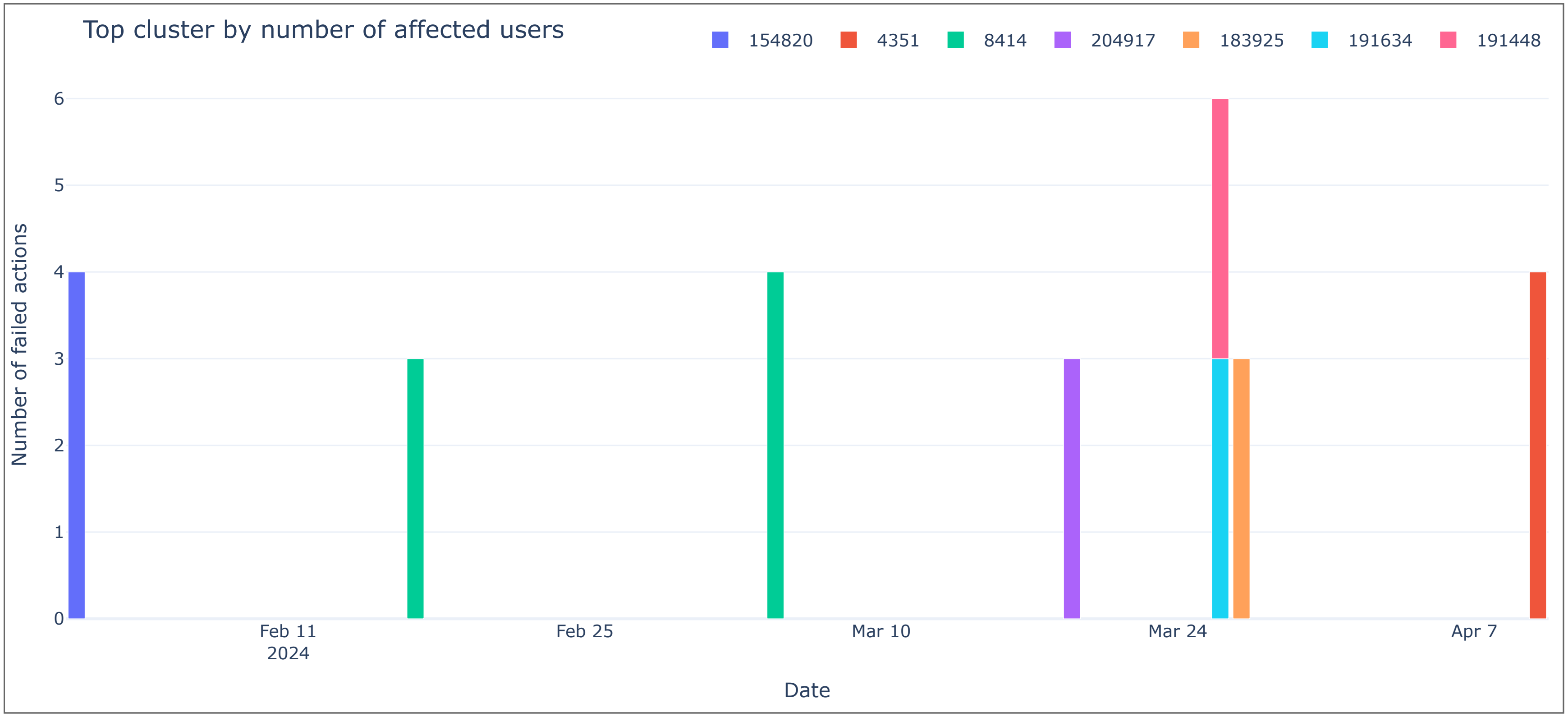
```
print(actions.loc[335630, "action_output_failure"])
```

```
fatal: reference is not a tree: f5b4060fec2bd637a88bfafbcc16b87f0241656c
```

Statistics: Top clusters by affected users

In [44]:

```
top_clusters_by_unique_users_fig.show()
```



Statistics: Top clusters by affected users

Outage in an external service:

In [46]:

```
print(actions.loc[4351, "action_output_failure"])
```

```
error An unexpected error occurred: "https://registry.yarnpkg.com/@babel/code-frame/
-/code-frame-7.12.13.tgz: Request failed \"502 Bad Gateway\"".
info If you think this is a bug, please open a bug report with the information provi
ded in "/home/circleci/project/yarn-error.log".
info Visit https://yarnpkg.com/en/docs/cli/install for documentation about this comm
and.
```

Lint error in a commonly edited file(s):

In [47]:

```
print(actions.loc[309408, "action_output_failure"])
```

```
  This project uses prettier to format all JavaScript code.
    Please run yarn prettier-all and add changes to files listed below to your commi
t:


packages/react-test-renderer/src/ReactTestRenderer.js
```

Statistics: Top clusters by affected users

Flaky test:

```
In [48]:
```

```
print("\n".join(actions.loc[8414, "action_output_failure"].splitlines()[1:16]))
```

```
ReferenceError: You are trying to access a property or method of the Jest environmen
t after it has been torn down. From packages/react-devtools-shared/src/__tests__/sto
reStressTestConcurrent-test.js.

      108 |
      109 |   if (recursivelyFlush) {
    > 110 |     while (jest.getTimerCount() > 0) {
          |                 ^
      111 |       await actDOM(async () => {
      112 |         await actTestRenderer(async () => {
      113 |           jest.runAllTimers();

      at actAsync (packages/react-devtools-shared/src/__tests__/utils.js:110:17)
      at _loop31 (packages/react-devtools-shared/src/__tests__/storeStressTestConcur
rent-test.js:1285:9)
      at _loop26 (packages/react-devtools-shared/src/__tests__/storeStressTestConcur
rent-test.js:1357:304)
      at Object.<anonymous> (packages/react-devtools-shared/src/__tests__/storeStres
sTestConcurrent-test.js:1358:282)
```

Integrations

1. **Incident Management** - widespread errors trigger incident management proc.
2. **Impact Assessment** - alerts have information about the number of affected users.
3. **Remediation Information** - Users can add remediation steps to error clusters.
4. **Error Suppression** - certain widespread errors can be suppressed and the the integration process can be resumed.
5. **Batch Retry** - once the root cause is resolved we can batch retry validation process for a precise set of blocked Code Reviews.
6. **Topline Metrics** - we estimate the impact of specific widespread errors on the topline metrics measuring developer productivity.

Summary

1. Clustering of near-identical documents eliminates the need of maintaining a long list of normalization rules.
2. Minhash and LSH Forest are efficient solutions for detecting near-identical documents. Proposed settings proposed density of ~2M cluster per 1 GB.
3. For the React case study, during the 1 year period we observed 350k erros. Proposed pipeline produced 300k unique error clusters taking 150MB.
4. Proposed solution scales well and can be deployed in large CI systems.

Appendix

Can you simply hash error message?

Performance of such solution depends heavily on the quality of extraction and normalization steps. In practice, for large-scale systems the maintenance cost becomes prohibitively high because this solution is:

- prone to extraction getting stale,
- prone to normalization getting stale,
- prone to changes in the codebase (e.g. new frame in call stack).