TNL-9545 - Elasticsearch Usage and Replacement

- Summary
- Usages
  - Courseware Indexer
    - Index #1: courseware_content
      - Document Structure
      - Queries Executed
      - Data Amounts
        - prod-edx
        - prod-edge
    - Index #2: library_index
      - Document Structure
      - Queries Executed
      - Data Amounts
        - prod-edx
        - prod-edge
    - Index #3: course_info
      - Document Structure
      - Queries Executed
      - Data Amounts
        - prod-edx
        - prod-edge
  - Blockstore Library Content Indexer
    - Index #1: content_library_block_index
      - Document Structure
      - Queries Executed
      - Data Amounts
        - prod-edx
        - prod-edge
    - Index #2: content_library_index
      - Document Structure
      - Queries Executed
      - Data Amounts
        - prod-edx
        - prod-edge
  - Course Teams Indexer
    - Index: course_team_index
      - Document Structure
      - Queries Executed
      - Data Amounts
        - prod-edx
        - prod-edge
- Raw Data
  - prod-edx
  - prod-edge
  - Script To Analyze
- Other Findings
  - Abstraction Leakage
  - Learning MFE - course content search not enabled
  - http://edx.org and edge.edx.org - do not use course info search
- Other Thoughts

Summary

As per this ticket:

TNL-9545 - Getting issue details... STATUS

the TNL team is performing discovery into the parts of the platform which the team owns that uses Elasticsearch (ES). Specifically, we own:

- edx-search usages within edx-platform
  - `cms/djangoapps/contentstore` - Courseware indexer
  - `openedx/core/djangoapps/content_libraries` - Blockstore-used library content indexer
  - `lms/djangoapps/teams` - Course teams indexer

edx-search provides a layer in front of ES which enables generic calls to be used when indexing content and querying that indexed content - instead of making direct ES calls.

Usages

Courseware Indexer

edx-platform code path: `cms/djangoapps/contentstore/courseware_index.py`

Index #1: `courseware_content`

Index used to search all the courseware content from all course blocks for all course runs.

Document Structure

```
{
    "course": <courserun ID>
    "org": <org from courserun ID above>
    "content_type": <block type such as Text, Sequence, etc.>
    "id": <usage ID>
    "start_date": <course start date>
    "content_groups": <content groups - dict of usage keys to group
names list values>
    "course_name": <course run ID>
    "location": <URL path to content>
    "content": <specific XBlock fields retrieved via block.
index_dictionary()>
}
```

Queries Executed

Arbitrary queries with filters:

- (optional) string to use in full text search of all content fields
- (optional) fields which must exist and must match a certain value
- (optional) fields which, if they exist, must match a certain value
- (optional) fields which, if they exist, must **not** match a certain value
- (optional) terms used to aggregate results within ES

Data Amounts

This index holds the most data **by far** than any of the other indices.

prod-edx

```
For index courseware_content:
  Number of documents: 18,091,780
  Total size of all documents (bytes): 22,178,956,562 (20.66 GB)
  Average document size (bytes): 1226
```

prod-edge

```
For index courseware_content:
  Number of documents: 8,842,392
  Total size of all documents (bytes): 8,199,948,232 (7.64 GB)
  Average document size (bytes): 927
```

Index #2: `library_index`

Index used to search all the v1 content library content, which lives in the modulestore. This index has been replaced by the blockstore library's content indexer `content_library_block_index` index.

Document Structure

```
{
    library:
    content_type:
    problem_types:
    "content": <specific XBlock fields retrieved via block.
index_dictionary()>
}
```

Queries Executed

N/A

Data Amounts

prod-edx

This index does **not** exist on prod-edx.

prod-edge

This index exists on prod-edge - but there's no documents stored in it.

```
For index library_index:
  Number of documents: 0
  Total size of all documents (bytes): 2,080 (2.03 KB)
```

Index #3: `course_info`

Index used to search all the course about blocks for all course runs.

Document Structure

```
{
    # non-'content' fields can filter by exact match only
    "course":
    "org":
    "advertised_start":
    "announcement":
    "start":
    "end":
    "effort":
    "title":
    "university":
    "number":
    "enrollment_start":
    "enrollment_end":
    "org":
    "modes":
    "language":
    "invitation_only":
    "catalog_visibility":
    # only 'content' field is analyzed by elasticsearch, and allows
text-search
    "content":
        "display_name":
        "overview":
        "title":
        "university":
        "number":
        "short_description":
        "description":
        "key_dates":
        "video":
        "course_staff_short":
        "course_staff_extended":
        "requirements":
        "syllabus":
        "textbook":
        "faq":
        "more_info":
        "ocw_links":
}
```

Queries Executed

Arbitrary queries with filters:

- (optional) string to use in full text search of all content fields
- (optional) fields which must exist and must match a certain value
- (optional) fields which, if they exist, must match a certain value
- (optional) fields which, if they exist, must **not** match a certain value
- (optional) terms used to aggregate results within ES

Data Amounts

prod-edx

```
For index course_info:
  Number of documents: 38,160
  Total size of all documents (bytes): 58,415,823 (55.71 MB)
  Average document size (bytes): 1531
```

prod-edge

```
For index course_info:
  Number of documents: 25,154
  Total size of all documents (bytes): 38,727,052 (36.93 MB)
  Average document size (bytes): 1540
```

Blockstore Library Content Indexer

edx-platform code path: `openedx/core/djangoapps/content_libraries/libraries_index.py`

The blockstore repo itself does **not** contain any code that interfaces with/uses Elasticsearch. The ES-interfacing code it uses is all in edx-platform at the path above.

This functionality was added with the development of Content Libraries v2, which has not launched yet in Open edX platform. Because of this, no data exists in either of the indices below in the http://edx.org production environments.

Index #1: `content_library_block_index`

Index used to search all the v1 content library content, which lives in the modulestore (MongoDB).

Document Structure

```
{
    "schema_version": <independent version number to bump when schema
changes>
    "id": <library ID>
    "library_key": <library usage key>
    "is_child": <true if block is a child of another block in the
library>
    "def_key": <definition key for the library>
    "display_name": <displayed name of the library block>
    "block_type": <type of the library block>
    "has_unpublished_changes": <true if library has unpublished changes>
    # only 'content' field is analyzed by elastisearch, and allows text-
search
    "content": {
        "id": str(item),
        "display_name": get_block_display_name(def_key),
    },
}
```

Queries Executed

- The only query filters on:
  - a single library key
  - is_child=False (top-level blocks only)
  - (optional) one or more block types
  - (optional) full text search against id and display_name

Data Amounts

prod-edx

None - has not launched.

prod-edge

None - has not launched.

Index #2: `content_library_index`

Index used to search all the v2 content library content, which lives in the blockstore (MySQL).

Document Structure

```
{
    "schema_version": <independent version number to bump when schema
changes>
    "id": <library ID>
    "uuid": <library UUID>
    "title": <library title>
    "description": <library description>
    "num_blocks": <number of blocks in the library>
    "version": <latest library version>
    "last_published": <datetime when last published>
    "has_unpublished_changes": <true if any library blocks are changed
but unpublished>
    "has_unpublished_deletes": <true if any library blocks are deleted
but unpublished>
    # only 'content' field is analyzed by elasticsearch, and allows
text-search
    "content": {
        "id": <library ID>
        "title": <library title>
        "description": <library description>
    },
}
```

Queries Executed

- The only query filters on:
  - one or more library keys
  - (optional) full text search against id, title, and description

Data Amounts

prod-edx

None - has not launched.

prod-edge

None - has not launched.

Course Teams Indexer

edx-platform code path: `lms/djangoapps/teams/search_indexes.py`

Index: `course_team_index`

Index used to search all the CourseTeams model instances.

Document Structure

```
{
    id:
    pk:
    discussion_topic_id:
    name:
    course_id:
    topic_id:
    date_created:
    description:
    country:
    language:
    last_activity_at:
    organization_protected:
    # only 'content' field is analyzed by elasticsearch, and allows
text-search
    content:
    {
        <course team name>
        <course team description>
        <course team country name>
        <course team language>
    }
}
```

Queries Executed

- The only query filters on:
    - course run ID
    - (optional) specific username used to only search teams in which the username is a member
    - (optional) topic ID
    - (optional) whether an organization is protected
    - (optional) full text search on name, description, country name, & language

Data Amounts

prod-edx

```
For index course_team_index:
  Number of documents: 11,134
  Total size of all documents (bytes): 14,127,778 (13.47 MB)
  Average document size (bytes): 1269
```

prod-edge

```
For index course_team_index:
  Number of documents: 600
  Total size of all documents (bytes): 718,115 (701.28 KB)
  Average document size (bytes): 1197
```

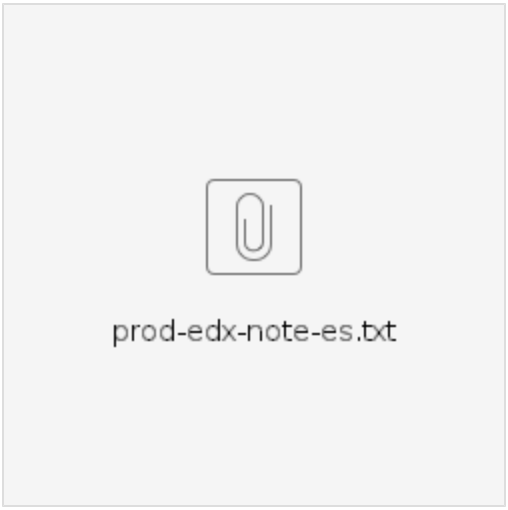
Raw Data

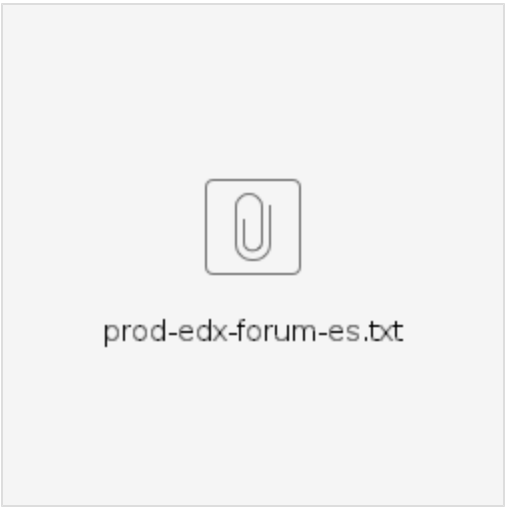
SRE recently executed this REST API call against the prod-edx and prod-edge ES clusters (with the correct host/port):

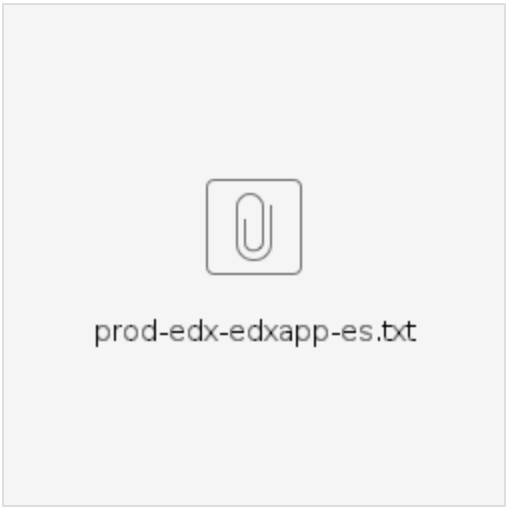
`curl -X GET "localhost:9200/_stats?pretty"`

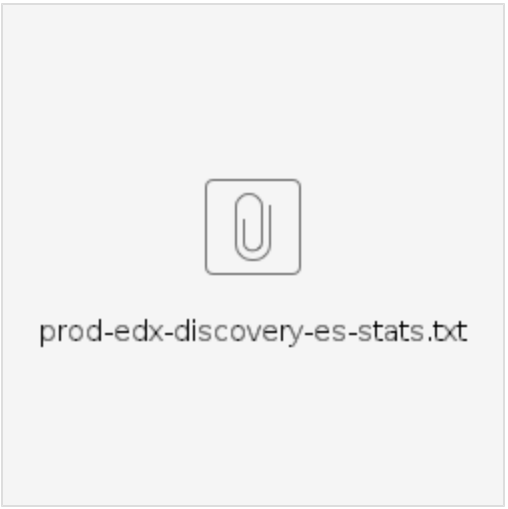
These stats provided the numbers about data storage above. I'll include the raw files below.

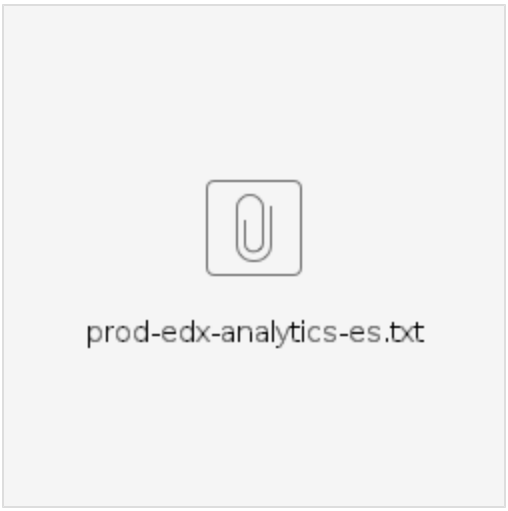
prod-edx


prod-edx-note-es.txt

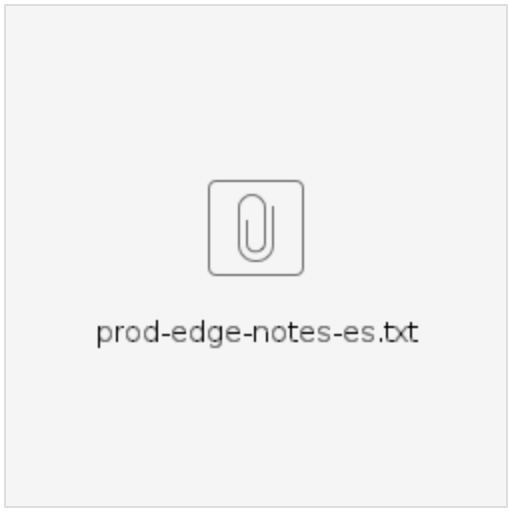

prod-edx-forum-es.txt

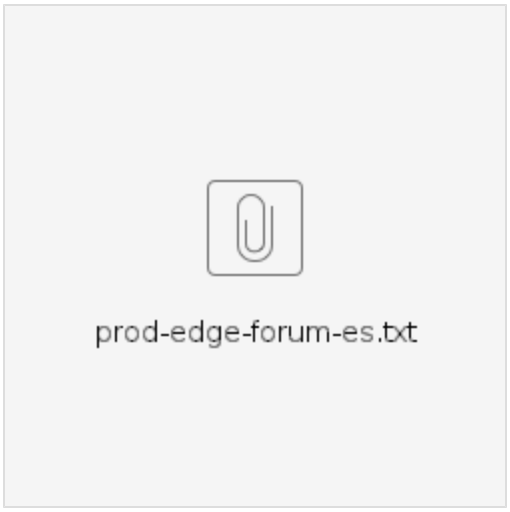

prod-edx-edxapp-es.txt

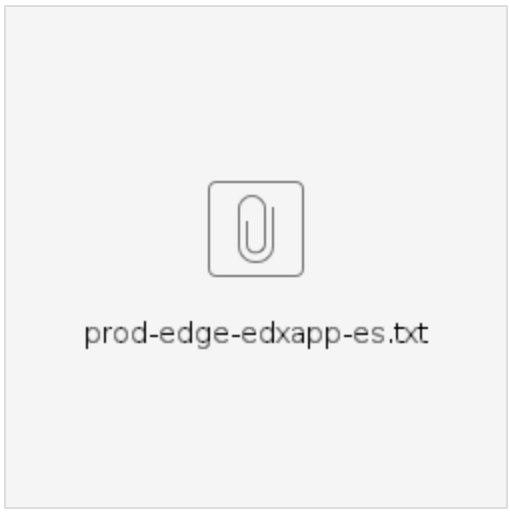

prod-edx-discovery-es-stats.txt

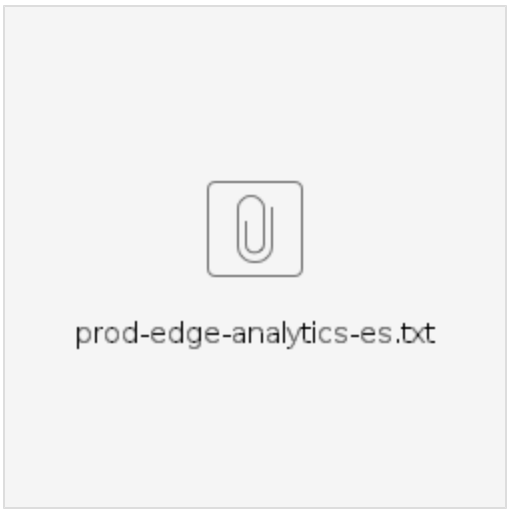

prod-edx-analytics-es.txt

prod-edge









Script To Analyze

I analyzed the JSON using the following Python script. Feel free to modify it for your own needs!

```
import io
import json
import logging
import pprint
import sys

import click

logging.basicConfig(stream=sys.stdout, level=logging.INFO)
LOGGER = logging.getLogger(__name__)


def humanbytes(B):
    """Return the given bytes as a human friendly KB, MB, GB, or TB
string."""
    B = float(B)
    KB = float(1024)
```

```
    MB = float(KB ** 2) # 1,048,576
    GB = float(KB ** 3) # 1,073,741,824
    TB = float(KB ** 4) # 1,099,511,627,776

    if B < KB:
        return '{0} {1}'.format(B,'Bytes' if 0 == B > 1 else 'Byte')
    elif KB <= B < MB:
        return '{0:.2f} KB'.format(B / KB)
    elif MB <= B < GB:
        return '{0:.2f} MB'.format(B / MB)
    elif GB <= B < TB:
        return '{0:.2f} GB'.format(B / GB)
    elif TB <= B:
        return '{0:.2f} TB'.format(B / TB)


@click.command()
@click.option(
    '--stats_path', '-s',
    required=True,
    help='Path to stats file output from Elasticsearch'
)
@click.option(
    '--es_index', '-i',
    multiple=True,
    help='Index name for which to get stats'
)
def extract_es_stats(stats_path, es_index):

    with open(stats_path, 'r') as stats_file:
        stats = json.load(stats_file)

    # stats_str = json.dumps(stats, indent=4)
    # LOGGER.info(stats_str)

    for index in es_index:
        try:
            index_data = stats['indices'][index]
        except:
            LOGGER.error("Index %s not found", index)
            continue
        print(f"For index {index}:")

        index_data_totals = index_data["total"]
        num_docs = index_data_totals["docs"]["count"]
        docs_size = index_data_totals["store"]["size_in_bytes"]

        print(f"  Number of documents: {num_docs:,}")
        print(f"  Total size of all documents (bytes): {docs_size:,}
({humanbytes(docs_size)})")
```

```
        if num_docs > 0:
            print(f"  Average document size (bytes): {docs_size /
num_docs:.0f}")
        print("\n")

if __name__ == "__main__":
    extract_es_stats() #pylint: disable=no-value-for-parameter
    sys.exit(0)
```

Other Findings

Abstraction Leakage

In `cms/djangoapps/contentstore/management/commands/reindex_course.py`, `searcher._es` is used several times. That member is the Elasticsearch implementation itself. edx-search provides an abstraction - a wrapper - around Elasticsearch for most usages but the abstraction leaks at this point. Whatever the eventual ES replacement is decided to be, the abstraction should be fixed in this file.

Learning MFE - course content search *not* enabled

The legacy LMS courseware offered the ES-backed feature to search the courseware index. However, this search was **not** added to the new learning MFE - and it still hasn't been added as a learning MFE feature as of March 2022. So - the existing prod-edx `courseware_content` index isn't currently being queried/used.

http://edx.org **and edge.edx.org** - do *not* use course info search

http://edx.org uses its own custom course discovery application. prod-edge uses the new course home page MFE which does *not* include course info search. So - the `course_info` index is used for course discovery on Open edX installations only.

Other Thoughts

- I did **not** perform an analysis of all the data that is stored in Elasticsearch in each index. It's entirely possible that some large percentage of it doesn't need to be searched. For example, does the `courseware_content` index contain content from the blocks of archived courses? Is a lot of the block content actually duplicated content from courses that were re-run?
    - General point: The amount of data stored in each index **might not** be the amount that would really need to be stored in a different text-searching data store. It'd be worth investigating that aspect.
- With the size of data stored in the courseware index (~20 GB) alone (not counting forums search and other usages), I'd be wary of using the MySQL full text and intermingling that data with the transactional data in our production MySQL instance.