Accessing WARC files via SQL

Sebastian Nagel

Common Crawl

Motivation

WARC files aren't easy to navigate and the CDX index isn't optimal to access WARC files by other criteria than URL or domain name.

Since spring 2018 Common Crawl provides a "columnar" index in "Parquet" format which can be queried and analyzed using SQL. It enables both users and web archive curators

- to gain insights into the archives and aggregate statistics and metrics within minutes
- pick captures by any provided metadata (e.g., content language, MIME type) to process the data "vertically" at scale

The columnar index has soon become the mostly used data format. At present, it contains over 100 billion rows covering all monthly crawls released since 2017 and occupies 7.5 TiB of storage.

Example 1: Languages Used on Spanish Web Sites

Which languages are used on web sites hosted under the `.es` top-level domain? This question is answered by the following SQL query and the columnar index:

```
SELECT COUNT(*) AS n_pages,
       COUNT(DISTINCT(url_host_registered_domain)) AS n_domains,
       content_languages
FROM "ccindex"
WHERE (crawl = 'CC-MAIN-2019-22'  -- April and May 2019 crawls
       OR crawl = 'CC-MAIN-2019-18')
  AND subset = 'warc'      -- only successful fetches
  AND url_host_tld = 'es' -- restrict to .es top-level domain
   -- skip pages with more than one language:
  AND NOT content_languages LIKE '%,%'
GROUP BY content_languages
HAVING COUNT(*) >= 10000
ORDER BY COUNT(*) DESC;
```

n_pages	domains	language
28966935	264430	`spa`
2054091	52278	`eng`
354378	6486	`cat` (Catalan)
133221	1118	`glg` (Galician)
61647	3298	`fra` (French)
40918	2318	`deu` (German)
31762	892	`eus` (Basque)
30863	119	`jpn`
25875	1007	`por`
22442	619	`rus`
16241	1086	`ita`
11553	361	`zho` (Chinese)

As expected, the most widely used language is Spanish. Other frequently used languages are: English, the minority languages of Spain (Catalan, Galician, Basque), neighboring languages (French and Portuguese) and those spoken by visitors arriving to Spain (German, Japanese).

How It Works

A Spark job converts the CDX index line by line mapping CDX fields to columns. URLs are split into components (domain name, path, etc.) to fill additional columns for fast aggregations and filtering on URL parts. Each of the 30 columns is defined in the table schema, e.g.,

```
{"name": "content_digest",
 "type": "string",
 "nullable": true,
 "metadata": {
  "description":
    "SHA-1 content digest (WARC-Payload-Digest)",
  "example": "CH7IV3XAD3M7A42JARKRLJ3T5PGGCGXD" }},
```

The schema is used to validate the values during write. The Parquet format integrates the schema and makes the table self-describing.

Table rows are sorted same as the CDX index by SURT URL – e.g., `com,example)/path/` – to optimize look-ups by domain name. Column values of the same domain also tend to be more homogeneous which has a positive impact on the data compression ratio.

The table is partitioned by monthly crawl and subset (successful fetches, 404s and redirects, robots.txt). Partitions organize the table files into subdirectories and allow incremental table updates and zero-cost filtering on partition columns.

About Common Crawl

Common Crawl is non-profit organization which regularly crawls a significant sample of the web and makes the data accessible free of charge to everyone interested in running machine-scale analysis on web data.

At present, we crawl every month up to 3.0 billion web pages. The data is hosted in the Amazon cloud as part of the AWS Open Data program.

Contact: https://commoncrawl.org/ sebastian@commoncrawl.org

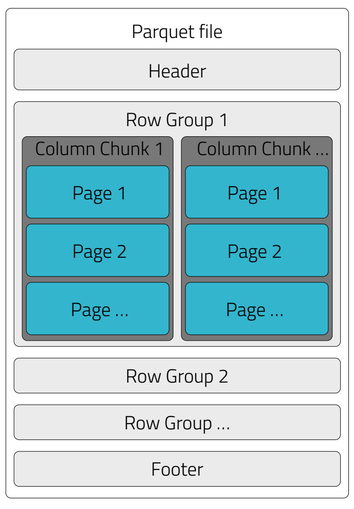
Parquet – A Columnar and Sustainable File Format

Parquet is an interchangeable but efficient column-oriented storage and file format:

- fully specified and documented, defined semantics, integrated schema (self-describing), guaranteed backward compatibility
- typed binary data representation for fast (de)serialization
- primitive and logical data types: boolean, integer, floating point number, string, date, etc.
- system and language independent API
- efficient to process – "read only what you need"

Data Layout – Rows and Columns

The table is split vertically into "row groups". The column values within one row group are hold in "column chunks".



A column chunk is split into "pages". Pages store the column values using a suitable encoding (plain, dictionary, run-length/bit-packed, delta). With dictionary encoding all values are hold in a separate dictionary page to speed up look-ups and filtering.

Pages are optionally compressed (gzip, zstd, etc.). Page-level compression (same as per-record WARC compression) allows to read only the requested pages.

The footer holds all information needed to "navigate" a Parquet file:

- the Parquet file format version
- the table schema and the number of rows
- metadata about row groups and column chunks

The metadata includes the location/offsets of row groups and column chunks. Additional statistics – the number of values (total, distinct, null) and min and max values – allow to skip entire row groups if a specific column value is not contained.

Example 2: WARC Storage Occupied per MIME Type

Common Crawl tries to crawl only HTML pages without page dependencies (images, CSS, JavaScript). However, a small percentage of non-HTML content is accepted to obtain a broad sample of document formats used on the web.

The issue with PDF documents, images and other non-HTML formats is that they tend to occupy more storage in WARC archives. But which formats at which scale?

```
-- average length and occupied storage of WARC records by MIME type
SELECT COUNT(*)                                              AS pages,
       round(COUNT(*)*100.0/SUM(COUNT(*)) OVER(), 3) AS perc_pages,
       round(AVG(warc_record_length)/power(2,10), 0) AS avg_rec_kB,
       round(SUM(warc_record_length)/power(2,40), 3) AS storage_TB,
       round(SUM(warc_record_length) * 100.0
          / SUM(SUM(warc_record_length)) OVER(), 3) AS perc_storage,
       content_mime_detected
FROM "ccindex"
WHERE crawl = 'CC-MAIN-2019-22' -- May 2019
  AND subset = 'warc'           -- only successful fetches
GROUP BY content_mime_detected
ORDER BY storage_TB DESC, n_pages DESC;
```

The SQL query above aggregates the WARC record length by the detected MIME type and calculates average and total sum. The result is sorted by the amount of occupied storage:

pages	%	avg.rec. kiB	storage TiB	%	MIME type
2033659795	75.890	17	32.012	65.019	text/html
605403020	22.592	15	8.290	16.837	application/xhtml+xml
19423997	0.725	388	7.014	14.246	application/pdf
4158147	0.155	257	0.997	2.024	image/jpeg
166558	0.006	885	0.137	0.279	audio/mpeg
633587	0.024	225	0.133	0.270	image/png
181213	0.007	484	0.082	0.166	application/zip
3944276	0.147	10	0.036	0.074	application/rss+xml
43070	0.002	847	0.034	0.069	video/mp4
42868	0.002	802	0.032	0.065	audio/mp4
38406	0.001	902	0.032	0.066	appl./vnd.android.package-archive
54795	0.002	499	0.025	0.052	application/epub+zip

Although the May 2019 dataset includes only 0.7% PDF files, these account for 7 TiB or 14% of the total storage. To minimize the storage usage we decided to increase the revisit frequency for storage-intensive formats.

Processing Engines and Frameworks

The following processing engines and big data frameworks have been successfully tested with the columnar index:

- Amazon Athena, a SQL query service to analyze data in Amazon S3. Athena builds on Presto, a distributed SQL query engine for big data
- Apache Hive, a data warehouse software project for managing large datasets residing in distributed storage using SQL. Queries are executed by MapReduce or Spark jobs
- Apache Spark, a general-purpose cluster-computing framework. Columnar data formats can be accessed through SQL or a dataframe API.

Example 3: Vertical Access to WARC Captures

Users of the Common Crawl frequently demand subsets of the data fitting their use case – all pages of a specific language, country or domain, only shopping-related or public sector sites, etc. However, WARC files are primarily organized by capture time and it would be hard or even impossible to organize them in a way so that all use cases are covered.

If the metadata in the columnar index allows to define a desired subset, it can be easily extracted from the archives using the indexed WARC filenames and record offsets to pick the WARC records via HTTP range requests. The Python code snippet below demonstrates how this procedure can be used to create a word frequency list from Icelandic web pages.

```
# "load" the columnar index (no actual load, only makes it available)
session = SparkSession.builder.getOrCreate()
df = spark.read.load('s3://commoncrawl/cc-index/table/cc-main/warc/')
df.createOrReplaceTempView('ccindex')
sqldf = spark.sql('SELECT url, warc_filename,  warc_record_offset,
                         warc_record_length
                  FROM "ccindex"
                  WHERE crawl = "CC-MAIN-2018-43"
                    AND subset = "warc"
                    AND content_languages = "isl"')
# alternatively load the result of Athena query
sqldf = session.read.format("csv").option("header", True) \
           .option("inferSchema", True).load(".../path/to/csv")

warc_recs = sqldf.select("url", "warc_filename", "warc_record_offset",
                         "warc_record_length").rdd

# simple Unicode-aware word tokenization (not suitable for CJK languages)
word_pattern = re.compile('\w+', re.UNICODE)

def fetch_process_warc_records(self, rows):
    """Fetch all WARC records defined by filenames and offsets in rows,
       parse the records and the contained HTML, split the text into words
       and emit pairs <word, 1>"""
    s3client = boto3.client('s3')

    for row in rows:
        url = row['url']
        warc_path = row['warc_filename']
        offset = int(row['warc_record_offset'])
        length = int(row['warc_record_length'])
        rangereq = 'bytes={}-{}'.format(offset, (offset+length-1))
        response = s3client.get_object(Bucket='commoncrawl',
                                       Key=warc_path,
                                       Range=rangereq)
        record_stream = BytesIO(response["Body"].read())
        for record in ArchiveIterator(record_stream):
            page = record.content_stream().read()
            text = html_to_text(page)
            words = map(lambda w: w.lower(), word_pattern.findall(text))
            for word in words:
                yield word, 1

word_counts = warc_recs.mapPartitions(fetch_process_warc_records) \
                       .reduceByKey(lambda a, b: a + b)
```

After running the Spark job you get the most frequent words in about one million Icelandic web pages:

22994307	*og*	6578254	*sem*	3893682	*2018*
19802034	*í*	6313945	*til*	3817965	*2*
15765245	*að*	5264266	*við*	3308209	*ekki*
15724978	*á*	4872877	*1*	3205015	*is*
8290840	*er*	4432790	*með*	3165578	*af*
8088372	*um*	4423975	*fyrir*	3051413	*en*

Links / Resources

https://github.com/commoncrawl/cc-index-table – Java project to convert the CDX index into the Parquet table. Includes examples of SQL queries and code to select WARC records by a SQL query and extract the captures into a WARC file

https://commoncrawl.org/2018/03/index-to-warc-files-and-urls-in-columnar-format/

https://github.com/commoncrawl/cc-pyspark – Python code to process Common Crawl data on Spark, optionally filtered via the columnar index, includes the code used for the Icelandic word count

https://parquet.apache.org/ – the Apache Parquet project

IIPC Web Archiving Conference, 6–7 June 2019, Zagreb, Croatia