Providing large trade data sets for research using Apache Parquet and R Shiny

Keywords: International Trade, Large Data, Apache Parquet, Apache Arrow, R

1. INTRODUCTION

As an economics research institute focused on international economics, we are interested in working with up-to-date international trade data with a fine level of granularity. In the early days this trade data was downloaded manually and stored either using Excel or as a comma-separated-values file on a simple file server, i.e. in a shared folder. With the evolution and increasing proliferation of the statistical programming language R, a simple automation for downloading and storing bulk files of trade data was set up. The resulting files were still comma-separated-values and Stata files, as most of our researchers are using Stata for econometric analysis. But producing static files based on csv or an XML-derivative (in the case of Stata) comes with the cost of file size, resulting in larger reading and writing times. This makes it hard to keep the data a-jour, and potentially reduces the possibility for the storage of e.g. different product classifications at a time, due to the sheer size of the data. Furthermore, large databases then introduce the need of saving data in one file per year or one file per exporter country. This makes queries that have to search (and thus load) across all files very costly and time intensive.

With the use of the R package *arrow* [1], which provides access to the functionality of Apache Parquet (a column-oriented data storage format with an efficient data compression), and enhanced in-memory performance through Apache Arrow, we are a) able to reduce the size of the data sets and b) improve the speed of queries notably, when extracting subsets of data from the database. This allows us to keep different data sets up to date using R-based cronjobs for downloading and ETL into an Apache Parquet database, using a small Debian-based server running R, RStudio Server and R Shiny Server. Additionally, we set up an R Shiny GUI for visually extracting data for downloading in various formats (Excel, csv, R, Stata), and incorporated a simple API using the R package *plumber* [2] for simple extraction of data and loading it into R and/or Stata using HTTP Requests.

1. METHODS

In the following sections we present the hardware in use, the workflow of our scheduled cronjob (for downloading data, ETL, etc), the basic structure of the Apache Parquet database set up, using R package arrow [1], as well as the GUI provided for the institute-wide users of international trade data.

1.1. Server Configuration

We are using Debian 10.6 64bit / Linux 4.19.0-11-amd64 running on a 4-core AMD A10-7800 Radeon R7 with 32GB RAM and a 200GB SSD, using R v4.0.2, RStudio Server v1.3.1093, and Shiny Server v1.5.14.948.

1.2. Cronjob structure

The nightly cronjob consists of the following R scripts which are executed subsequently:

1. Check for data updates using provided API (UN Comtrade), automated look-up in folder (EU Comext)
2. Download new and/or updated bulk data provided as archives (zip, 7z)
3. Download new ad-hoc correspondences (EU Comext)
4. ETL to Apache Parquet for provided classifications
5. Generate new classifications using new/own correspondences
6. Compute aggregates and complement data sets with data from other sources
7. Load into Apache Parquet database

1.3. Apache Parquet

Apache Parquet is a column-oriented data storage format provided in the R package *arrow*, providing efficient data compression and encoding schemes with enhanced performance to handle complex and large data in bulk. The *arrow* package additionally provides the functionality of Apache Arrow, a cross-language development platform for handling flat and hierarchical in-memory data, organized for efficient analytic operations on modern hardware like CPUs and GPUs at large scale (see François et al. [1]). With the implementation of both concepts, we expect 1) a substantial reduction in file size, 2) a performance boost in loading data, and 3) a reduction in time when handling large in-memory data.

The Apache Parquet database consists of many small files, depending on the type of partition that is chosen when setting it up. In our case, a database on international trade in goods, we chose partitioning by (1) Product classification, (2) Year, (3) Trade Flow, and (4) Reporter, as this is from our experience the most common and efficient way of querying the database.

1.4. Graphical User Interface & API

After the cronjob of the R scripts presented above is finished, the updated database is available via an R Shiny [3] GUI using your favourite web browser, and via HTTP requests provided by the *plumber* API. Figure 1 below shows the home page of the GUI, offering the possibility to choose between databases and pre-compiled datasets, e.g. as visible in Figure 1 a pre-compiled NACE Rev1 2-digit data set based on the Eurostat COMEXT database, using a correspondence of the WIOD project [4].

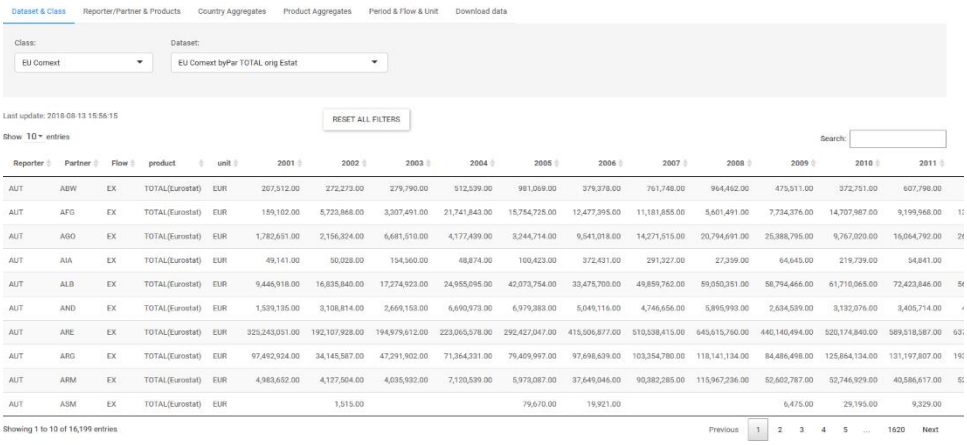

Figure 1. GUI - Start Screen

The GUI allows for country- and product-specific filtering. The available options are dynamically generated as the displayed tables differ in their possible choices of countries and products. Additionally, we offer a user-based construction of country- and product-

aggregates. This is implemented using the fast-aggregation features of the *data.table* [5] package. With the use of R package *eurostat* [6], up-to-date currency exchange rates are fetched from the Eurostat database, allowing the conversion between EUR and USD using period average/end of period rates. Figure 2 shows the possibility to filter trade flows (Export, Import), as well as the time period of interest for the resulting dataset.

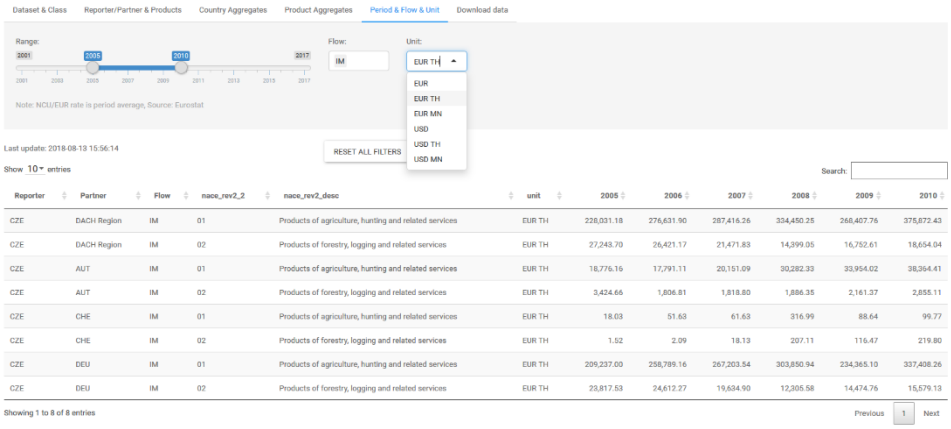

Figure 2. GUI - Filtering Time, Flow & Unit

As researchers in our institute are using different software to analyse the resulting datasets, different options for the export are provided using packages *haven* (Stata), *openxlsx* (Excel), *data.table* (fast csv output) and R base functionality.

2. RESULTS

Compression-wise we were able to reduce the size of the data sets for all product classifications the UN Comtrade trade database massively from ~900GB to 55GB (compression ratio of ~16 or saving in space of 93.89%). This is even significantly higher as reported in a paper by Boufea et al. (2017) [7] who indicated compression ratios of around 10 compared to tab-separated-values files.

We now compare the performance of the pre-existing database consisting of comma-separated-values files with the performance of the columnar-based Apache Parquet format, based on queries usually used at our institute. For the purpose of demonstration, we are using data based on the UN Comtrade database in its HS96 classification. The following queries are practical use cases and try to show different variations of performance based on the different need of loading large amounts of data into the memory before sub-setting the data. Loading and sub-setting is done with R package *data.table* when using csv files. Table 1 shows the resulting performance measured in seconds and minutes, as well as the resulting improvement in percent respectively.

Table 1. Query performance

Query	Parquet	CSV	Improvement
Reporter = AUT, Partner = DEU, Year = 2017, product = TOTAL, flow = Export	3.3”	9.7”	66%
Reporter = AUT, Partner = DEU, Year = 2008-2018, product = TOTAL, flow = Export	25.4”	1.7’	75%
Reporter = EU28, Partner = USA, Year = 2017, product = TOTAL, flow = Export	4.0”	9.7”	35%
Reporter = EU28, Partner = (USA, JPN, KOR, RUS, BRA), Year = 2017, product = TOTAL, flow = Export & Import	7.3”	21.5”	66%
Reporter = EU28, Partner = (USA, JPN, KOR, RUS, BRA), Year = 2008-2018, product = TOTAL, flow = Export & Import	53.5”	3.5’	75%

3. CONCLUSIONS

This paper demonstrates that the Apache Arrow framework outperforms the csv-based database in all tested queries. The mean improvement in elapsed time is 63%. Another advantage of the Apache Arrow architecture is that it hides the distributed file system and enables the user to write easier understandable code. The csv-based database makes it necessary to write loop constructs that are less performant and more elaborate to program. Furthermore, the Apache Parquet data storage format reduces file sizes dramatically, allowing for cheaper storage of large and even big data sets on modern computer hardware. In the presented use case, file size was drastically reduced by over 93%, resulting in an impressive compression factor of 16 compared to a database based on comma-separated-values files.

3.1. Outlook

We see the parquet data as a basis, since it is kept up to date. From this basis we can then easily expand our database in several ways, e.g. 1) extend the database by including detailed tariff data, as well as data on the occurrence of non-tariff measures (NTMs). The inclusion of this data would require incorporating non-reported zero trade flows. First tests show that this would increase storage by a factor of ~4, and 2) include computation of often-used aggregations, such as bilateral totals or similar.

4. REFERENCES

[1] Romain François, Jeroen Ooms, Neal Richardson and Apache Arrow (2020). arrow: Integration to 'Apache' 'Arrow'. R package version 1.0.1. https://CRAN.R-project.org/package=arrow

[2] Barret Schloerke and Jeff Allen (2020). plumber: An API Generator for R. R package version 1.0.0. https://CRAN.R-project.org/package=plumber

[3] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2020). shiny: Web Application Framework for R. R package version 1.5.0. https://CRAN.R-project.org/package=shiny

[4] Timmer, M., Erumban, A.A., Gouma, R., Los, B., Temurshoev, U., de Vries, G.J., Arto, I.A., Genty, V.A.A., Neuwahl, F., Francois, J. and Pindyuk, O., 2012. The world input-output database (WIOD): contents, sources and methods (No. 20120401). Institute for International and Development Economics.

[5] Matt Dowle and Arun Srinivasan (2020). data.table: Extension of `data.frame`. R package version 1.13.0. https://CRAN.R-project.org/package=data.table

[6] Leo Lahti, Janne Huovari, Markus Kainu, Przemyslaw Biecek. Retrieval and analysis of Eurostat open data with the eurostat package. R Journal 9(1):385-392, 2017. Version 3.6.1 Package URL: http://ropengov.github.io/eurostat Manuscript URL: https://journal.r-project.org/archive/2017/RJ-2017-019/index.html

[7] Boufea, A., Finkers, R., van Kaauwen, M., Kramer, M. and Athanasiadis, I.N., 2017, December. Managing variant calling files the big data way: Using HDFS and apache parquet. In Proceedings of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (pp. 219-226).