EasyChair Preprint

№ 9225

Application Architecture for Obtaining Data From Scientometric Databases

Mykhailo Hunko, Vitalii Tkachov, Oleksii Liashenko and Jan Rabčan

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

November 3, 2022

Application Architecture For Obtaining Data From Scientometric Databases

Mykhailo Hunko
Department of Electronic Computers
Kharkiv National University Of Radio Electonics
Kharkiv, Ukraine
mykhailo.hunko@nure.ua

Vitalii Tkachov
Department of Electronic Computers
Kharkiv National University Of Radio Electonics
Kharkiv, Ukraine
tkachov@ieee.org

Oleksii Liashenko
Department of Electronic Computers
Kharkiv National University Of Radio Electonics
Kharkiv, Ukraine
oleksii.liashenko@nure.ua

Jan Rabčan
Department of Informatics
University of Žilina
Žilina, Slovakia
jan.rabcan@fri.uniza.sk

***Abstract*—Scientometric data from bibliographic and reference databases with a limited user interface are used to analyze the effectiveness of the scientific staff, educational and educational institutions in the field of science. Automation of the process of obtaining the author's scientometric indicators from bibliographic and reference databases through a limited user interface could accelerate the creation of a picture of the scientometric activity of the institution's employees. The main goal of quickly obtaining the author's scientometric indicators from bibliographic and abstract databases through a limited user interface is a significant saving of human resources during the manual analysis of these data. Also in this case, the possibility of errors during manual data collection is minimized. Data collection can be done in a variety of ways and means. Among those considered in the article: data parsing, receiving data through the application program interface and receiving data through third-party software solutions supplied by companies. The paper considers approaches to obtaining and processing these data.**

Keywords— Scientometric data, collecting data, processing, handler, architecture

I. Introduction

Scientometric data help to compare the scientific achievements of an employee of a higher educational institution, help applicants to decide on the choice of an educational institution. International rankings, such as the QS World University Rankings [1], when building rankings of educational institutions, also consider the performance of researchers from a given university. For example, the mentioned rating considers data from the Scopus database. Scopus - it is one of the components of the integrated science and information environment SciVerse [2]. Let us go to the future the base will cover over 50 million abstracts. Scopus Rubricator (ASJK) main thematic sections, divided into 335 subsections, and polythematic articles are indexed in several sections. The index base includes 18,000 names of scientific knowledge in technical, medical, and humanitarian fields. Database of indexes of scientific journals, materials of conferences and serial books. SciVerse Scopus has been dispersed and owned by Elsevier Publishing Corporation. The database is available for a fee through a web interface. The Scopus search system is integrated with the Scirus search system for web site search and the patent data base.

The Scopus database is positioned by Elsevier Publishing Corporation as the largest universal abstract database in the world with the ability to track scientific publication citations. According to the announced strategy, this database should become the most comprehensive and complete resource for searching scientific literature. The novelty of the work is the proposed method for obtaining scientometric data and comparing the speed of the multithreaded approach.

The purpose of this work is to review the architecture for obtaining data from scientometric and abstract databases.

II. Existing solutions overview

A. Getting data throw API

The easiest approach to get data is getting data by API. Many of the services that provide access to data usually require a fee for their service. Or they have a limited number of free inquiries after which they require payment. In this case, it is appropriate to consider the mechanism of tracking the number of requests by the services and user identification on their side.

We can identify the user and check the number of requests available to him using an API key. This is a kind of key-identifier. An API key is an identifier assigned to an API client that is used to authenticate the application calling the API. Typically, this is a unique alphanumeric string that is included in an API call that the API receives and validates. Many APIs use a key to track usage and detect bad or malicious requests.

The API usually requires developers to purchase a key before making requests. The process should be documented on the API developer's website and tell you everything you need to get started.

In most cases, you will need to create a developer account with your email and other information. You will then be prompted to register the project and enter any information about the project that API owners need to know [4].

B. Getting data by parsing

Obtaining scientometric indicators of authors from bibliographic and reference databases with a limited user interface can be carried out by parsing the HTML markup of web-oriented services of bibliographic and reference databases [7][10]. The Selenium WebDriver open-source

software library can be used as a software driver. The output is an author ID in a database with a limited user interface (Fig. 1). The main advantage of this approach is in independence from any third-party restrictions. This approach may be limited only by the hardware capabilities of a particular user.

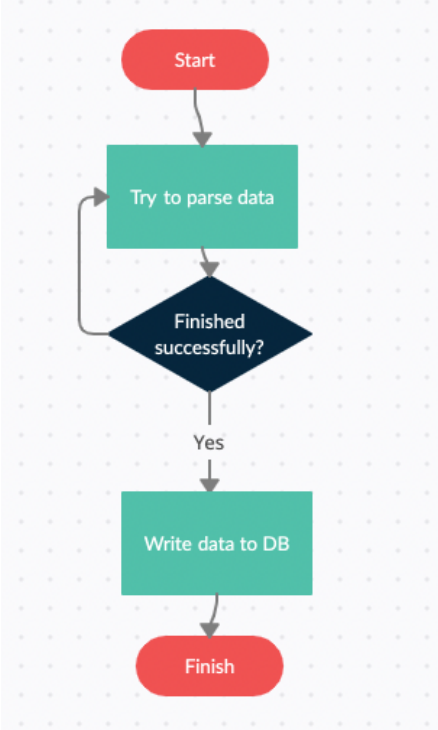

Fig. 1 Block diagram of parsing data.

C. *Getting data from providers*

This approach greatly simplifies the data acquisition process. It also provides a large selection of data forms. For inexperienced developers, this can be a significant simplification, as it ignores the need to subscribe and configure the environment to obtain data directly from scientometric databases.

III. Application Archeticture Development

Let us present the abstract model in the form of several layers:

- Data source layer.
- Handler layer.
- Data layer.

Data source layer is a layer of data that is supplied to us from scientometric databases. In this case, the model is an abstract. In the implementation, this can be a json-formatted string, or an HTML page, or any kind of supplied data, including serialized and non-serialized data. Usually, this layer can't be edited by developer, because it was developed by data provider.

The data processing layer is the layer in which the processing received from a particular implementation takes place. This can be web page parsing, data deserialization, as well as any other data processing option, depending on the data format that is presented in the data source layer. Also, data processing can take place both in real time and with a "lazy" start, which will process data only on demand or after a specified period of time. This approach can reduce the load on the data processing layer hardware. It is also possible to use a multi-threaded implementation of data processing if it is necessary to speed up the processing time. In this case, it is worth noting that regardless of the chosen programming language, creating and deleting threads is a laborious and time-consuming process.

SQL is a non-procedural database language focused on a large amount of information to be processed. Non-procedural means that it primarily focuses on what data to call, delete, or insert, rather than how to do it. Quantity-oriented means that with the help of this language it is possible to process significant amounts of information in groups. This type of presentation is also beneficial for large amounts of data, because the database can index the information that is stored inside. Indexes are special tables that can be used by a database search engine (hereinafter referred to as the DB) to speed up data retrieval. You just need to add an index pointer to the table. An index in a database is very similar to an index at the end of a book. An index helps speed up data retrieval queries (SELECT [WHERE]) but slows down the process of adding and modifying records (INSERT, UPDATE). Indexes can be added or removed without affecting the data itself.

First, let us look at the model for obtaining data through the API. To obtain data in this way, ones need to obtain an API key and then request data (Fig. 2)

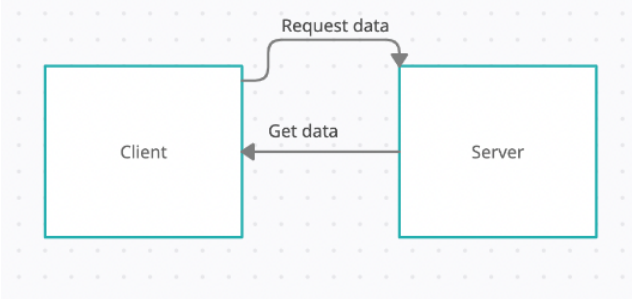

Fig. 2 Getting data scheme.

After receiving the data, the data are processed in the processing layer. The processing layer can be represented a virtual machine, or a set of virtual applications, or a hybrid solution. It performs a request filtering procedure and, in the case of a normal request, redirects it to the correct resource server.

When processing information and using a multi-threaded environment, the choice of balance between threads should be carried out according to:

$$I = P / n + P \tag{1}$$

where P is thread load, n is a number of requests in the queue.

Thus, the processing of information will occur in the shortest possible time. In the event of an execution error on one of the threads, execution must be queued and executed according to the distribution from the queue. Such a system helps to minimize the likelihood of a fatal error.

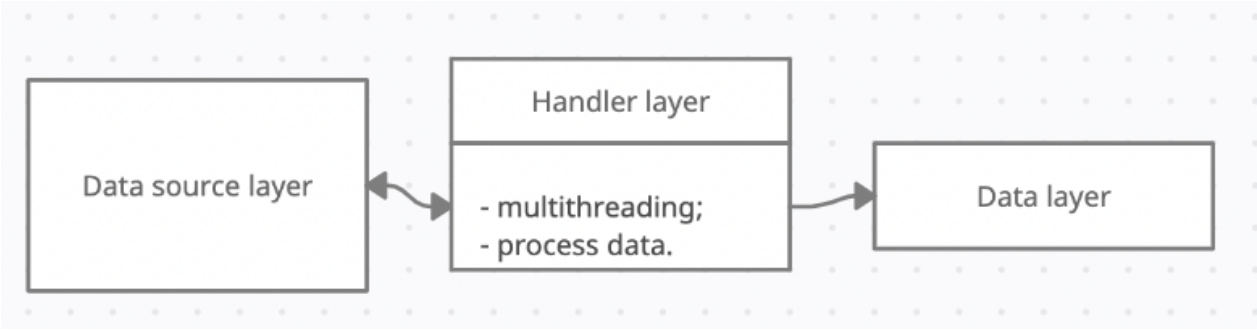

Fig. 3 Layer architecture.

In case of data layer, it would be better to use SQL database. When modeling a database, consider the following:

- Determine what the table is for and what its structure is. Therefore, the first thing to decide is to determine the purpose for the database. What type of data is the database for? Will the user only work with records and need to pay attention to transactions, or is he more interested in data analytics? Where should the base be deployed? Will it track the behavior of customers or just manage the relationship between them?
- What data to choose for storage? It is always necessary to clearly understand why and for what purposes a particular value is stored in the database and choose the correct data format.
- Model data with care. If the entity is very difficult to classify in one word or phrase, then it's time to use subtypes (child entities). If the entity leads its own life, has attributes that describe its behavior and its appearance, as well as relationships with other objects, then you can safely use not only a subtype, but also a supertype (the parent entity). If you ignore this rule, other developers will get confused in your model and will not fully understand the data and the rules for how to collect it.
- Use the correct data types. Using the wrong data type can result in less accurate data, difficulty joining tables, synchronizing attributes, and bloated file sizes. To ensure the integrity of the information, an attribute should only contain data types that it accepts. If age is entered into the database, then make sure that the column stores integers with a maximum of 3 digits.
- Prefer natural. It is best to use a natural, or business, key (natural key). It has semantic meaning, so you avoid duplication in the database. If only the business key is not unique (first name, last name, position) and is repeated in different rows of the table or it must be changed, then the generated artificial, surrogate key (artificial key) should be assigned as the primary key.

So, the relationship between the layers should look like shown in Fig. 3. Data source layer and handler layer communicate between each other because handler layer requests data and data source layer returns this data. And data layer only gets data without any requests. Data layer does not know about other layers.

IV. Testing the Proposed Architecture

To test the mathematical model of the distribution of the load on the threads, mathematical modeling was created, and the results of data processing were measured depending on the number of threads and the time spent per unit of processing.

TABLE I. Runtime Comparison

Data amount	Threads number	Execution time, ms	Acceleration
15	1	T_1= 238 680	-
30		T_1= 477 792	-
60		T_1= 953 320	-
100		T_1= 1 427 691	-
15	2	T_2= 149 410	$S=T_1/T_2$= 1.60
30		T_2= 299 451	$S=T_1/T_2$= 1.59
60		T_2= 596 186	$S=T_1/T_2$= 1.60
100		T_2= 892 216	$S=T_1/T_2$= 1.61
15	4	T_4= 96 790	$S=T_1/T_4$= 2.47
30		T_4= 193 148	$S=T_1/T_4$= 2.36
60		T_4= 383 704	$S=T_1/T_4$=2.48
100		T_4= 573 431	$S=T_1/T_4$= 2.49
15	8	T_8= 73 184	$S=T_1/T_8$= 3.26
30		T_8= 142 046	$S=T_1/T_8$= 3.36
60		T_8= 293 081	$S=T_1/T_8$= 3.25
100		T_8=431 706	$S=T_1/T_8$= 3.31

For testing, data sets consisting of 15, 30, 60 and 100 elements were taken, and mathematical experiments were carried out with 1, 2, 4 and 8 streams. As it can be seen from Table 1, the most efficient acceleration can be achieved when distributed over 8 threads. For the most accurate time measurement, milliseconds were used. It is also worth noting that the time spent on creating threads was not taken into account when calculating the time. Simulations have shown that creating threads can take anywhere from 3 percent to 7 percent of the total time it takes to complete the entire operation for a full dataset, which is quite high. The

percentage can be reduced by increasing the number of processed elements, because creating threads will take the same time, and the total processing time will increase due to the increased volume.

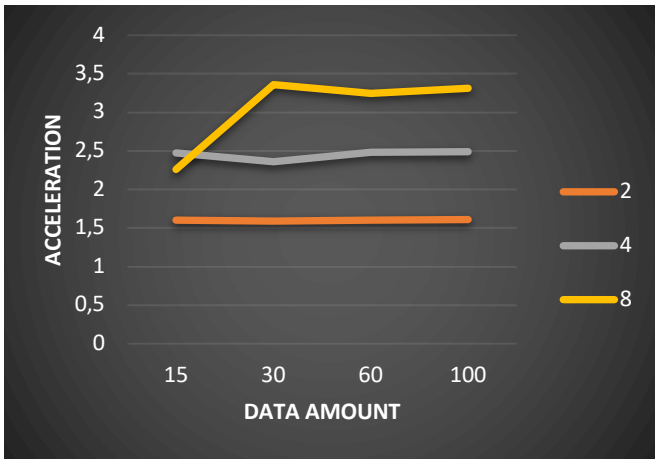

Fig. 4 Comparison table of acceleration.

Fig. 4 shows the dependence of the acceleration of execution on the amount of data. The graph shows appropriate dependencies for 2, 4, 8 threads. After analyzing this graph, we can confidently state that with an increase in threads, the productivity of the software, as a whole, increases. However, it should be kept in mind that an infinite increase in flows can only lead to additional loads. The real effective maximum allowable number is measured in terms of the number of cores on a particular computer or virtual machine.

V. Conclusions

As a result of the work carried out, the principles of the architecture and the application module for obtaining scientometric data were considered. In particular:

- The positive aspects of using a three-layer architecture are given
- The behavior of the multi-threaded implementation of the proposed architecture is modeled.
- An analysis of the importance of obtaining scientometric databases was carried out.

The test shows that increasing the number of threads increases the acceleration, but not linearly. With the computational power it makes sense to increase the number of threads involved.

For further research, it is proposed to study the increase in the fault tolerance of the proposed architecture, including by transferring some layers to cloud services

VI. Acknowledgment

This publication has been produced with the support of the Integrated Infrastructure Operational Program for the project: Creation of a Digital Biobank to Support the Systemic Public Research Infrastructure, ITMS: 313011AFG4, co-financed by the European Regional Development Fund. The mathematical models have been created under grant ‘‘Mathematical Models based on Boolean and Multiple-Valued Logics in Risk and Safety Analysis’’ (reg.no. SK-FR-19-0003) by the Slovak Research and Development Agency.

References

[1] World University Ratings by Subject 2021, [online] Available: https://www.topuniversities.com/sites/default/files/Subject%202021%20v4.pdf.

[2] Aleksandr Spivakovsky, Maksym Vinnyk, Maksym Poltoratskiy, Yulia Tarasich, , Kateryna Panova, Anton Melnychenko, Development of rating systems for scientometric indices of universities, ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer 2018, vol. 2104, 2018, pp. 420-430.

[3] Scientometric databases. National library of Ukraine, [online] Available: http://www.nbuv.gov.ua/node/1367

[4] Spivakovsky, A., Vinnyk, M., Tarasich, Y., Poltoratskiy M.: Design and development of information system of scientific activity indicators. . Ermolayev, V., Spivakovsky, A., Nikitchenko, M., Ginige, A., Mayr, H. C., Plexousakis, D., Zholtkevych, G., Burov, O., Kharchenko, V., and Kobets, V. (Eds.): ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer. Proc. 12th Int. Conf. ICTERI 2016, vol. 1614, pp. 103-110. CEUR-WS.org (2016).

[5] Zaitseva E., Levashenko V., Construction of a reliability structure function based on uncertain data, IEEE Transactions on Reliability, vol. 65, no. 4, 2016, pp. 1710 – 1723.

[6] Levashenko V., Zaitseva E., Puuronen S., Fuzzy classifier based on fuzzy decision tree, Proc. of the Int. Conf. on Computer as a Tool (EUROCON), September 9-12, 2007, Warsaw, Poland, pp. 823 – 827, doi: 10.1109/EURCON.2007.4400614.

[7] Mukhamediev R., Popova Y., Kuchin Y., Zaitseva E., Kalimoldayev A., Symagulov A., Levashenko V., Abdoldina F., Gopejenko V., Yakunin, K., et al., Review of Artificial Intelligence and Machine Learning Technologies: Classification, Restrictions, Opportunities and Challenges. Mathematics 2022, 10, 2552. doi: 10.3390/math10152552

[8] O. Barkovska, N. Axak, D. Rosinskiy and S. Liashenko, "Application of mydriasis identification methods in parental control systems," 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT), Kiev, 2018, pp. 459-463.

[9] Fesenko, T., Ruban, I., Karpenko, K., Fesenko, G., Kovalenko, A., Yakunin, A., Fesenko, H. Improving of the decision-making model in the processes of external quality assurance of higher education. Eastern-European Journal of Enterprise Technologies. 2022. Vol. 1 (3 (115)), P. 74–85. doi: https://doi.org/10.15587/1729-4061.2022.253351

[10] Ruban, Igor, et al. "Segmentation of the Images Obtained From Onboard Optoelectronic Surveillance Systems by the Evolutionary Method." Eastern-European Journal of Enterprise Technologies, vol. 5, no. 9, 2017, pp. 49-57, doi:10.15587/1729-4061.2017.109904