Сучасний стан наукових досліджень та технологій в промисловості. 2021. № 1 (15) ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

UDC 681.3.06 DOI: https://doi.org/10.30837/ITSSI.2021.15.053

D. ZOLOTARIOV

THE MECHANISM FOR CREATION OF EVENT-DRIVEN APPLICATIONS BASED ON WOLFRAM MATHEMATICA AND APACHE KAFKA

The article is devoted to the research and development of the mechanism of interaction between Wolfram Mathematica programs and Apache Kafka queue to provide the ability to build event-driven applications based on it. The **subject** of the research is the practical principles of building a mechanism for interaction between Wolfram Mathematica and Apache Kafka. The **purpose** of the article is to develop and substantiate practical recommendations regarding the formation of a mechanism for publishing messages to the Apache Kafka queue and reading messages from it for programs of the mathematical processor Wolfram Mathematica, which will make it possible to build event-driven applications. **Tasks**: to determine the mechanism of such interaction, prove the choice of tools for its implementation, create and test the obtained results. The research used the following **tools**: Apache Kafka, Kafkacat, the method of developing the Wolfram Mathematica package. The **results** of the research: the mechanism of interaction between Wolfram Mathematica and Apache Kafka was determined and the corresponding toolkit was created on its basis in the form of two Mathematica packages, which are built on using Apache Kafka as a queue client and third-party Kafkacat software, respectively. It is shown that the first option is less reliable and consumes much more computer resources during operation. It has been demonstrated that the Mathematica processor is currently not suitable in its pure form for real-time data analysis. Recommendations are given regarding the use of built-in compilation functions to increase the speed of such processing. **Conclusions**. Practical recommendations have been developed and substantiated regarding the formation of the mechanism of interaction between the Wolfram Mathematica mathematical processor and the Apache Kafka queue manager for the possibility of working in two directions with the queue: publishing messages and reading them. A toolkit for such interaction in the form of Mathematica packages has been created, their capabilities have been demonstrated, as well as comparison with each other. The economic benefit of using the described tools is shown. Future ways of its improvement are given.

Keywords: event-driven applications; queue manager; mathematical processor; saving resources and funds; Kafka; Mathematica.

Introduction

Recent years have been marked by the rapid development of micro service architecture and the distributed processing of information in real or near real time. This is primarily due to the development and improvement of data delivery mechanisms such as queue managers.

Based on them, such event-driven products are built as: IoT [1-2], which work with the flow of messages from "smart" things; web platforms [3] that display the results of receiving or processing events to the end user; data processing pipelines [4-7], responding to events of different nature, and others. The peculiarity of such systems is that they focus not so much on reducing the processing time of each individual event and bringing it closer to real time, but on the guaranteed and clear sequence of interconnected events and guarantees the processing of each of them. The latter comes to the fore because event generators and consumers are often completely technologically and algorithmically independent, and can also be located in space at a considerable distance from each other. In addition, consumers - queue data points - can appear in the system as needed, change at any time, and be excluded from it when they are no longer needed. The consequence of this feature is that the number of queue clients is limited only by the capacity of the queue managers' servers.

One of the most popular, fault-tolerant and powerful queue managers at the moment is Apache Kafka. It allows you to build a distributed system of brokers (managers) of the queue, which is able not only to dynamically adapt to the load from the queue customers of both types (generators and consumers), but also easily scale both vertically and horizontally.

The benefits of event-oriented architecture are increasingly being appreciated beyond the development of commercial products. One of the promising areas is the processing of such a message flow by mathematical processors, which allows you to use their full range of tools for analysis. For example, the implementation of the study of data obtained from Kafka in the mathematical package MathWorks MATLAB is already underway [8] and has a fairly rich functionality.

The Wolfram Mathematica processor is one of the world leaders in the field of symbolic and numerical data processing and is used in almost every field of knowledge and science, which is clearly seen, for example, in publications [9-12], where this matpacket is used to solve different areas of applied technology. Therefore, the construction of event-driven products based on it is an urgent task.

But there is still no effective and reliable mechanism for connecting this math processor and Kafka to receive or publish data. There is a single undocumented MQTTLink package [13] for the MQTT protocol (mosquitto), which, through third-party software, allows you to connect to a Kafka cluster. But it does not guarantee correct operation [14].

Therefore, the **purpose** of this article is to develop and substantiate practical recommendations for the formation of the mechanism of interaction between the mathematical processor Wolfram Mathematica and the queue manager Apache Kafka to work in two directions: publishing messages in the queue and consuming messages from it – which will build event-driven applications in Mathematica. The **task** of the article is to identify the necessary elements of such a mechanism and justify the choice of tools for their construction.

Using Apache Kafka

The Mathematica mat package itself, as mentioned above, does not have built-in interoperability with Kafka. And the easiest way to organize this interaction is to locally install Apache Kafka software on the client computer as a queue client and interact with the queue through it. The diagram of the system constructed on such approach is given in fig. 1. It should be noted that Kafka also requires the installation of a Java code execution platform. It should also be borne in mind that currently Kafka developers do not guarantee the correct operation of all components of their platform on Windows [15].

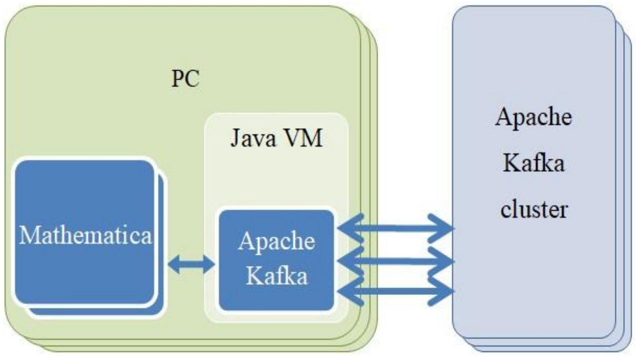


Fig. 1. Connecting to a Kafka cluster via a local Apache Kafka installation as a client

Scripts from the "/ bin /" subdirectory for Linux systems and "/ bin / windows /" for Windows systems in the Kafka installation directory are used for publishing, reading messages, and other operations in turn. For certainty, we will consider only Windows as a client, for Linux the difference will be only in the extension of scripts ".sh".

The publishing operation is a one-time execution of the instant termination script "kafka-console-producer.bat", which is performed each time to send a separate message to the queue, which is read from the standard input stream (stdin). Read operation - one-time execution of the script "kafka-console-consumer.bat" without completion, which is an endless process of reading messages from the queue and output them to the standard output stream (stdout). The main parameters of all scripts are: "--bootstrap-server" - a comma-separated list of boostrap servers to connect to the Kafka cluster, and "--topic" - the name of a specific queue. Hereinafter, these general parameters in the code are replaced by three dots for brevity.

Therefore, the construction of the mechanism of interaction of Mathematica with the Kafka cluster should be based on the use of built-in functions of executing external applications and reading streaming data that work with standard input and output streams.

To run external applications from a Mathematica document, use the Run function, the only parameter of which is the command to execute. The function of publishing in turn built on its basis looks as follows.

```
Run["echo \""<> key <>":"<> value <>"\" | kafka-console-producer.bat … "
  <>" --property \"parse.key=true\" --property \"key.separator=:\""];
```

The publication is made in the form of "key: value", which is indicated by two arguments of the script with the prefix "--property". Moreover, the "value" should not have time transfers. The function is a call to the echo environment command, which provides data for publication to the Kafka script input.

Creating and deleting queues is done in a similar way, for example, creating:

```
Run["kafka-topics.bat … --create"];
```

To read from a data stream in Mathematica, use the Read function, the parameters of which are the file name and read mode: string, write, word, and others. But if you put an exclamation point at the beginning of the file name, it will be treated as a command to execute and the Read function will return its output to the standard stream. Also, instead of the file name, you can pass the handle of the thread opened by the OpenRead function, which takes in the file name with all the comments above. The mechanism of operation of the Read function is such that it does not end until the data flow reaches the value of EndOfFile, i.e. does not end. The reading of queue data based on it is given below.

```
stream = OpenRead["!kafka-console-consumer.bat … --from-beginning"];
While[True,{
    Check[str=Read[stream,String], Break[];];
    If[str===EndOfFile || callback[str]===False, Break[];]
}];
Close[stream];
```

The above construction opens a data stream initiated by the script "kafka-console-consumer.bat" and reads in an infinite loop. Each message is processed by a user function in the callback variable. In this case, the simplest way to break messages is used – as a text string by hyphenation of the line "\ n". But if the messages contain such characters, it is possible to switch the reading mode to "Record" and set in the RecordSeparators option of the Read function a valid character for dividing the flow into messages. The use of the intermediate function OpenRead is intended to obtain a flow descriptor for its correct closure by the function Close, however, as mentioned above, you can pass the command directly by the first argument to the function Read. In the latter case, the thread will be closed only when the current Mathematica kernel is closed.

For ease of distribution and use, the developed functions based on the above are designed in the form of Mathematica package "KafkaLocalLink", which is built in a standard way and has the following structure of files and directories, shown in fig.2.

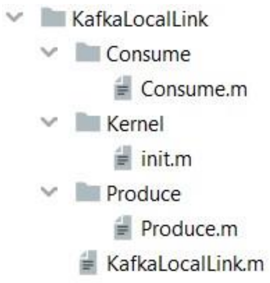


Fig. 2. The structure of the directories of the Mathematica package of interaction with Kafka

Each of the Consume.m and Produce.m files has descriptions of the function of obtaining a list of queue names and their specialized ones. The first is reading, the second is deleting, creating a queue, and publishing to it.

The advantages of separating the developed program code into a package include the possibility of its dynamic loading in Mathematica.

Using Kafkacat

An alternative approach to publishing to the Kafka queue might be to install Kafkacat [16] on the client, a third-party software for connecting to a Kafka cluster that does not require the Java platform but requires Windows version 10 with Windows Subsystem for installed and configured. Linux (WSL) [17] based on Ubuntu. The diagram of the system built on the use of Kafkacat is shown in fig. 3.

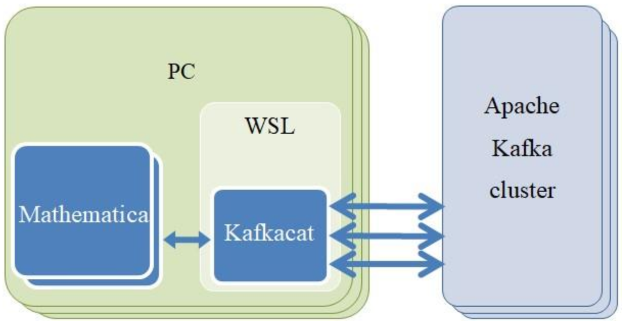


Fig. 3. Connecting to a cluster via a local Kafkacat installation

In terms of built-in features, this application is not inferior to the original Apache Kafka software, but much more convenient to use.

A similar package for Mathematica is based on it. Which differs only in the following remarks. The publishing team in the queue will look much shorter:

```
"… | kafkacat –P -K: -b … -t …"
```

The argument "-P" includes publishing mode, "-K" - sets the colon as a separator between the key and the value of the message, the last "-b" and "-t" - are completely similar to "--bootstrap-server" and "—topic".

The reading command from the queue will look like this:

```
"!kafkacat -C –J -u –q -b … -t …"
```

Where the argument "-C" includes read mode, "-J" - receive a message in full JSON-format, "-u" - switches to unbuffered output mode, "-q" - excludes the output of service information.

As mentioned above, using the "-J" parameter for Kafkacat returns a string that is a JSON object with complete information about the message from the queue: publication time, key, values, headers, offsets, and more. But Mathematica does not have a built-in convenient mechanism for working with the JSON format, so to process this string and get from it the fields "key" (message key) and "payload" (message text) you need to use the following approach:

```
json = ImportString[str, "JSON"];
key  = "key" /. json;
```

Similar constructs are used to retrieve other fields from the object, such as "headers", "offset" or "ts"..

Work demonstration and comparison

To implement the experiment, two Mathematica documents were created on the client computer: a generator and a reader. Each of them has its own core of the mathematical processor, which is allocated its own core of the CPU - for guaranteed parallel and independent document processing. The first developed package was used to connect to the Apache Kafka cluster. After receiving the results, the queue was deleted and the second Kafkacat-based package was used.

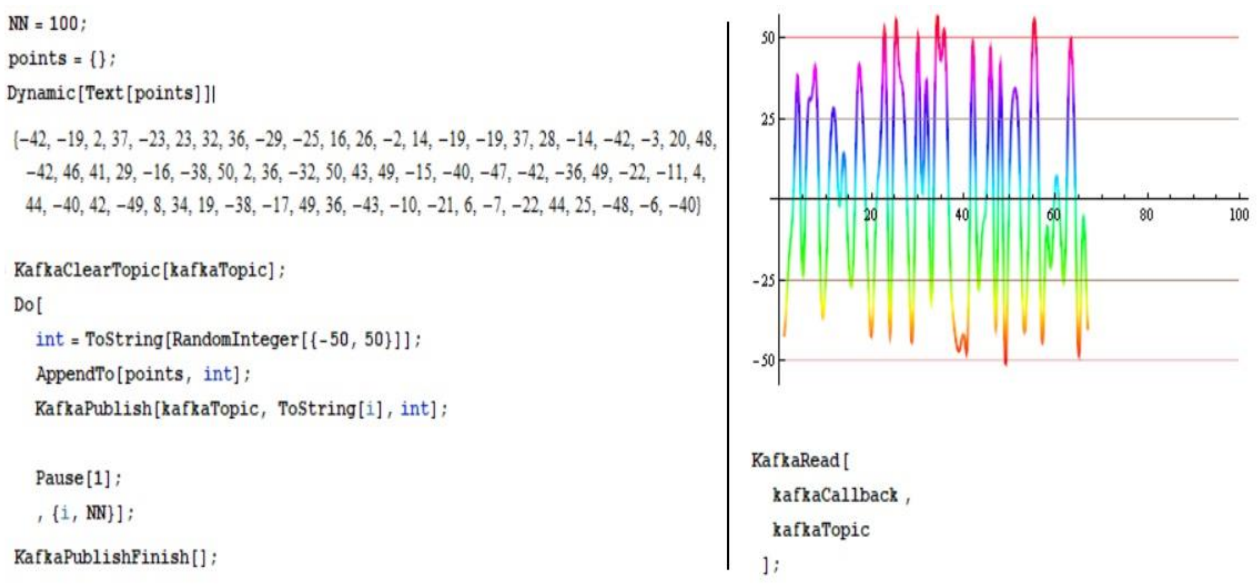
As a platform for the deployment of the queue, cloud technologies DigitalOcean were chosen, which is one of the world leaders, where servers based on Ubuntu 20.04 LTS x64 OS were located, which proved to be a reliable and fast platform in previous developments [20].

To demonstrate the operation of the developed tools for building event-driven programs in Mathematica, the generation of an arbitrary number in the range [-50.50] is selected, repeated 100 times with a delay of one second. A small number of iterations are chosen for the convenience of plotting.

The content of the experiment is as follows: on the publication side, an arbitrary number is generated and added to the end of the points array, which has a dynamic update (via the Dynamic function) in the document, and is sent to the queue. At the same time, the client-reader receives a message with this number, adds it to its array, which automatically leads to the restructuring of the graph based on it, which also has a dynamic output. The graph is constructed by the ListPlot function for the entire definition area [1,100] and with the option

"InterpolationOrder -> 2", which at the boundary points leads to the output values from the original range of values [-50.50] and serves as an additional load on message processing.

Since the client can only connect to a queue that already exists, the first to run was a document with the publication of messages, which first creates a queue, followed by a reader document. The result obtained at the same time for both documents for the package with Kafkacat is shown in fig. 4.



```
NN = 100;
points = {};
Dynamic[Text[points]]

{-42, -19, 2, 37, -23, 23, 32, 36, -29, -25, 16, 26, -2, 14, -19, -19, 37, 28, -14, -42, -3, 20, 48,
 -42, 46, 41, 29, -16, -38, 50, 2, 36, -32, 50, 43, 49, -15, -40, -47, -42, -36, 49, -22, -11, 4,
 44, -40, 42, -49, 8, 34, 19, -38, -17, 49, 36, -43, -10, -21, 6, -7, -22, 44, 25, -48, -6, -40}

KafkaClearTopic[kafkaTopic];
Do[
  int = ToString[RandomInteger[{-50, 50}]];
  AppendTo[points, int];
  KafkaPublish[kafkaTopic, ToString[i], int];

  Pause[1];
  , {i, NN}];
KafkaPublishFinish[];
```

```
KafkaRead[
  kafkaCallback,
  kafkaTopic
 ];
```

Fig. 4. Simultaneous publishing (right) and reading from the same queue (left) using the Kafkacat-based package

The figure above shows that the client-reader does not lag behind the client-generator. But in real time it is clear that such a lag is still there, although insignificant. The main delay that has been shown in the simulation is the updating of dynamic objects in the document: for the text field when publishing - imperceptible, for graphics - about 0.2s, which is visible to the eye. With more complex processing operations, the delays will be even greater.

This in turn indicates that the Mathematica package is not yet designed for real-time data analysis and cannot handle critical operations. For complex data stream analysis tasks, it is recommended to use in Mathematica compilation of functions in WVM bytecode or even C, which in practice [21] can increase its execution speed from 2 to 10 times or more depending on the input function being compiled.

The comparison of the speed of the developed packages is as follows. For publication, only the publication of data in the queue is selected as the most resource-intensive process. The experiment described above was repeated 15 times for each developed package to exclude measurement error. The measurement results were averaged. They are given in table 1.

Table 1. *Consumption of computer resources by clients of the Kafka queue*

Name	Using OM (MB)	CPU time usage (%)
Package based on Apache Kafka	132	79,3
Package based on Kafkacat	5	0,5

This clearly shows that Kafkacat is much faster to perform and more economical to operate, because it requires almost an order of magnitude less computer resources.

Prospects for further development of the system

Prospects for further development of the developed tools are the following improvements.

Due to the fact that the queue never ends and because of this, the program will never receive an EndOfFile signal during normal operation, it seems promising to develop an additional mechanism for forcibly exiting queue messages by the client-reader on a signal from the client-generator or in any other way, as well as to include in the package processing service fields "headers" from a JSON object describing the message in Kafkacat.

Disadvantages of this tool include the installation and configuration of multiple components on the Kafka cluster client computer. In addition, a direct connection to the Kafka cluster is associated with the following complications - each client needs to know the IP addresses of bootstrap servers, have authorization data on all servers in the cluster and know all the necessary settings to work with them. Therefore, it seems promising to transfer this component of the mechanism of interaction from cluster clients to a separate server, which will act as an intermediary, hiding all the settings of interaction with the cluster and authentication.

Conclusions

The paper develops and substantiates practical recommendations for the formation of the mechanism of interaction of the mathematical processor Wolfram Mathematica and the queue manager Apache Kafka for the possibility of working in two directions: publishing messages in the queue and consuming messages from it.

Appropriate tools have been created in the form of two Mathematica packages, built on the use of Apache Kafka as a queue client and third-party Kafkacat software, respectively.

It is shown that the first option is less reliable and requires much more machine resources during operation.

It has been demonstrated that the Mathematica processor is not currently suitable in its pure form for real-time queue data analysis. Recommendations for using built-in compilation features to increase message processing speed are given.

The economic benefit of using the described tools is achieved due to the possibility of developing powerful data analyzers from the Kafka queue with a relatively easy-to-learn Mathematica processor instead of developing narrowly specialized tools. And also due to fast and flexible updating of such program in a matpacket which can be changed at any moment and started at once on execution.

References

1. Ed-daoudy, A., Maalmi, K. (2019), "A new Internet of Things architecture for real-time prediction of various diseases using machine learning on big data environment", *Journal of Big Data*, Vol. 6, No. 104. DOI: https://doi.org/10.1186/s40537-019-0271-7
2. Mahapatra, T. (2020), "Composing high-level stream processing pipelines", Journal of Big Data, Vol. 7, No. 81. DOI: https://doi.org/10.1186/s40537-020-00353-2
3. Nasiri, H., Nasehi, S., Goudarzi, M. (2019), "Evaluation of distributed stream processing frameworks for IoT applications in Smart Cities", *Journal of Big Data*, Vol. 6, No. 52. DOI: https://doi.org/10.1186/s40537-019-0215-2
4. Jung, S., Kim, Y., Hwang, E. (2018), "Real-time car tracking system based on surveillance videos", *EURASIP Journal on Image and Video Processing*, Vol. 2018, No. 133. DOI: https://doi.org/10.1186/s13640-018-0374-7
5. Ismail, A., Truong, H. L., Kastner, W. (2019), "Manufacturing process data analysis pipelines: a requirements analysis and survey", *Journal of Big Data*, Vol. 6, No. 1. DOI: https://doi.org/10.1186/s40537-018-0162-3
6. Kim, Y. K., Kim, Y., Jeong, C. S. (2018), "RIDE: real-time massive image processing platform on distributed environment", *EURASIP Journal on Image and Video Processing*, Vol. 2018, No. 39. DOI: https://doi.org/10.1186/s13640-018-0279-5
7. Kolajo, T., Daramola, O., Adebiyi, A. (2019), "Big data stream analysis: a systematic literature review", *Journal of Big Data*, Vol. 6, No. 47. DOI: https://doi.org/10.1186/s40537-019-0210-7
8. GitHub (2020), "Mathworks-ref-arch/matlab-apache-kafka: MATLAB Interface for Apache Kafka", available at : https://github.com/mathworks-ref-arch/matlab-apache-kafka (last accessed 10 December 2020).
9. Rehman, S., Idrees, M., Shah, R. A. et al. (2019), "Suction/injection effects on an unsteady MHD Casson thin film flow with slip and uniform thickness over a stretching sheet along variable flow properties", *Boundary Value Problems*, Vol. 2019, No. 26. DOI: https://doi.org/10.1186/s13661-019-1133-0
10. Ghorbani, M. A., Singh, V. P., Sivakumar, B. et al. (2017), "Probability distribution functions for unit hydrographs with optimization using genetic algorithm", *Applied Water Science*, Vol. 7, P. 663–676. DOI: https://doi.org/10.1007/s13201-015-0278-y
11. DeCanio, S. J. (2020), "Can an AI learn political theory?", *AI Perspectives*, Vol. 2, Article 3. DOI: https://doi.org/10.1186/s42467-020-00007-2
12. You, X., Chen, D. R. (2018), "A new sequence convergent to Euler–Mascheroni constant", *Journal of Inequalities and Applications*, Vol. 2018, Article 75. DOI: https://doi.org/10.1186/s13660-018-1670-6
13. Mathematica Stack Exchange (2020), "Networking - Connect Mathematica to message broker - Kafka, NATS or mosquitto", available at: https://mathematica.stackexchange.com/questions/199848/connect-mathematica-to-message-broker-kafka-nats-or-mosquitto (last accessed 10 December 2020).
14. Mathematica Stack Exchange (2020), "Networking - MQTTLink TopicSubscribe[] cannot receive messages", available at: https://mathematica.stackexchange.com/questions/211940/mqttlink-topicsubscribe-cannot-receive-messages (last accessed 10 December 2020).
15. Apache Kafka (2020), "Apache Kafka", available at: https://kafka.apache.org/documentation/#os (last accessed 10 December 2020).
16. GitHub (2020), "Edenhill/kafkacat: Generic command line non-JVM Apache Kafka producer and consumer", available at: https://github.com/edenhill/kafkacat (last accessed 10 December 2020).
17. Microsoft Docs (2020), "An overview on the Windows Subsystem for Linux", available at: https://docs.microsoft.com/en-us/windows/wsl/ (last accessed 10 December 2020).
18. Stack Overflow (2020), "Kafka bootstrap-servers vs zookeeper in kafka-console-consumer", available at: https://stackoverflow.com/questions/41774446/kafka-bootstrap-servers-vs-zookeeper-in-kafka-console-consumer (last accessed 10 December 2020).
19. Stack Overflow (2020), "Apache kafka - bootstrap-server vs zookeeper params in consumer console", available at: https://stackoverflow.com/questions/53954877/bootstrap-server-vs-zookeeper-params-in-consumer-console (last accessed 10 December 2020).
20. Zolotariov, D. (2020), "The distributed system of automated computing based on cloud infrastructure", *Innovative Technologies and Scientific Solutions for Industries*, No. 4 (14), P. 47–55. DOI: https://doi.org/10.30837/ITSSI.2020.14.047
21. Zolotariov, D. A. (2020), "Automation and optimization of scientific and engineering calculations in Wolfram Mathematica", Kharkiv : FOP Panov A. M. ISBN: 978-617-7859-36-8 [In Ukrainian].

Received 23.12.2020

Відомості про авторів / Сведения об авторах / About the Authors

Золотарьов Денис Олексійович – кандидат фізико-математичних наук, Харків, Україна; email: denis@zolotariov.org.ua, ORCID: https://orcid.org/0000-0003-4907-7810.

Золотарёв Денис Алексеевич – кандидат физико-математических наук, Харьков, Украина.

Zolotariov Denis – PhD (Physics and Mathematics Sciences), Kharkiv, Ukraine.

РОЗРОБКА МЕХАНІЗМУ ДЛЯ СТВОРЕННЯ КЕРОВАНИХ ПОДІЯМИ ДОДАТКІВ НА БАЗІ WOLFRAM MATHEMATICA ТА APACHE KAFKA

Стаття присвячена дослідженню та розробці механізму взаємодії програм Wolfram Mathematica із менеджером черги Apache Kafka для надання можливості побудови на його основі керованих подіями додатків. **Предметом** дослідження є практичні засади побудови механізму взаємодії Wolfram Mathematica із Apache Kafka. **Метою** статті є розробка та обґрунтування практичних рекомендацій щодо формування механізму публікації повідомлень у чергу Apache Kafka та споживання повідомлень із неї для програм математичного процесору Wolfram Mathematica, що дасть можливість побудови керованих подіями додатків. **Завдання** роботи: визначити механізм такої взаємодії, обґрунтувати вибір інструментів для його реалізації, створити та протестувати отриманий результат. У ході дослідження використано засоби: інформаційні технології Apache Kafka, Kafkacat, спосіб побудови пакету Wolfram Mathematica. **Результати** дослідження: визначений механізм взаємодії Wolfram Mathematica із Apache Kafka та створений відповідний інструментарій на його основі у вигляді двох пакетів Mathematica, що побудовані на використанні Apache Kafka у якості клієнта черги та стороннього програмного забезпечення Kafkacat відповідно. Показано, що перший варіант є менш надійним та потребує значно більше машинних ресурсів під час роботи. Продемонстровано, що на даний момент математичний процесор Mathematica не підходить у чистому вигляді для аналізу даних у реальному часі. Дані рекомендації щодо використання вбудованих функцій компілювання для підвищення швидкості обробки. **Висновки**. Розроблені та обґрунтовані практичні рекомендації щодо формування механізму взаємодії математичного процесору Wolfram Mathematica та менеджеру черги Apache Kafka для можливості роботи у двох напрямках із чергою: публікації повідомлень та їх читання. Створений інструментарій для такої взаємодії у вигляді пакетів Mathematica, продемонстровані їх можливості, а також порівняння між собою. Показана економічна вигода від використання описаного інструментарію. Наведені майбутні шляхи його вдосконалення.

Ключові слова: керовані подіями додатки; менеджери черги; математичний процесор; економія ресурсів та коштів; Kafka; Mathematica.

РАЗРАБОТКА МЕХАНИЗМА ДЛЯ СОЗДАНИЯ УПРАВЛЯЕМЫХ СОБЫТИЯМИ ПРИЛОЖЕНИЙ НА БАЗЕ WOLFRAM MATHEMATICA И APACHE KAFKA

Статья посвящена исследованию и разработке механизма взаимодействия программ Wolfram Mathematica с менеджером очереди Apache Kafka для предоставления возможности построения на его основе управляемых событиями приложений. **Предметом** исследования являются практические принципы построения механизма взаимодействия Wolfram Mathematica с Apache Kafka. **Целью** статьи является разработка и обоснования практических рекомендаций относительно формирования механизма публикации сообщений в очередь Apache Kafka и чтения сообщений из нее для программ математического процессора Wolfram Mathematica, что даст возможность построения управляемых событиями приложений. **Задача** работы: определить механизм такого взаимодействия, обосновать выбор инструментов для его реализации, создать и протестировать полученный результат. В ходе исследования использованы **средства**: информационные технологии Apache Kafka, Kafkacat, способ разработки пакета Wolfram Mathematica. **Результат** исследования: определен механизм взаимодействия Wolfram Mathematica с Apache Kafka и создан соответствующий инструментарий на его основе в виде двух пакетов Mathematica, которые построены на использовании Apache Kafka в качестве клиента очереди и стороннего программного обеспечения Kafkacat соответственно. Показано, что первый вариант менее надежен и потребляет значительно больше компьютерных ресурсов во время работы. Продемонстрировано, что на данный момент математический процессор Mathematica не подходит в чистом виде для анализа данных в реальном времени. Даны рекомендации относительно использования встроенных функций компилирования для повышения скорости такой обработки. **Выводы.** Разработаны и обоснованы практические рекомендации относительно формирования механизма взаимодействия математического процессора Wolfram Mathematica и менеджера очереди Apache Kafka для возможности работы в двух направлениях с очередью: публикации сообщений и их чтения. Созданы инструментарий для такого взаимодействия в виде пакетов Mathematica, продемонстрированы их возможности, а также сравнение между собой. Показана экономическая выгода от использования описанного инструментария. Приведены будущие пути его усовершенствования.

Ключевые слова: управляемые событиями приложения; менеджеры очереди; математический процессор; экономия ресурсов и средств; Kafka; Mathematica.

Бібліографічні описи / Bibliographic descriptions

Золотарьов Д. О. Розробка механізму для створення керованих подіями додатків на базі Wolfram Mathematica та Apache Kafka. *Сучасний стан наукових досліджень та технологій в промисловості*. 2021. № 1 (15). С. 53–58. DOI: https://doi.org/10.30837/ITSSI.2021.15.053

Zolotariov, D. (2021), "The mechanism for creation of event-driven applications based on Wolfram Mathematica and Apache Kafka", *Innovative Technologies and Scientific Solutions for Industries,* No. 1 (15), P. 53–58. DOI: https://doi.org/10.30837/ITSSI.2021.15.053