# Dataflow adapter: a tool for integrating legacy applications into distributed stream processing

Alexey Podkoritov, Ksenia Repina and Gleb Radchenko
*School of Electrical Engineering and Computer Science*
*South Ural State University*
Chelyabinsk, Russia
alexeipodkorytov65@gmail.com, repinakv@susu.ru,
gleb.radchenko@susu.ru

Andrey Tchernykh
*Computer Science Department*
*CICESE Research Center*
Ensenada, Mexico
*South Ural State University*
Chelyabinsk, Russia
chernykh@cicese.mx

**In this paper, we propose the concept of the adapter, which permits to process data streams transmitted through Apache Kafka using legacy locally executed applications. Furthermore, every application handles input and output data streams like data files in the local file system and can be developed in any programming language or software framework without including the complexity of the distributed stream processing.**

**Keywords—stream processing, distributed computing, streaming platform, Apache Kafka**

## I. Introduction

With the rapid development of "Smart Manufacturing", "Industry 4.0" and "Industrial Internet of Things" technologies [1], the distributed processing of data streams from embedded machine controllers and sensors is gaining in importance. A common approach is based on using specialized streaming platforms as a middleware between various data processing applications [2]. An example of such platform is Apache Kafka [3] designed to transfer large amounts of data in highly loaded systems.

Apache Kafka's usage is based upon producer-consumer pattern, where data producers and consumers are implemented by the same applications that perform data processing. We propose to handle all the interactions with Apache Kafka via the separate application called the *dataflow adapter*. It allows performing data processing as though all the data are located in the local file system, which provides the following advantages over the traditional way:

*1) Data processing applications are independent of Apache Kafka.* It is not necessary for applications to interact with the streaming platform, so they can be developed in any programming language or software framework despite the fact there are no compatible or up-to-date Kafka client libraries.

*2) Local applications are integrated into the distributed system.* The adapter allows using existing data analysis, engineering and scientific software, designed to work with local data files, for distributed processing of data streams from Apache Kafka.

*3) The configurable interaction between applications.* It's difficult to manage interactions through Apache Kafka created by a large number of applications because every application controls data producers, consumer, and queues according to its own settings. All the dataflow adapter instances are controlled via different parts of the shared adapter configuration, which allows the centralized management of data interchange.

The reported paper was supported by RFBR, research project No. 18-07-01224 and by Act 211 Government of the Russian Federation, contract № 02.A03.21.0011

*4) Real-time access to data processing statistics.* The adapter measures the speed and the latency of data processing, the duration of data transfer between the application and the streaming platform and other performance metrics, which transmits through Apache Kafka for the further display and analysis.

## II. Related solutions analysis

When developing the adapter concept, we studied the popular cloud platforms Predix and Confluent, which feature polar opposite approaches to the implementation of distributed computing. In Predix, each application for data processing provides a standard API, which is called by the platform, and also receives and returns data of a strictly defined structure described in the configuration [4]. Confluent does not regulate the interface provided by the application or the structure of the data being processed, but applications need to use Apache Kafka through special libraries [5]. Thus, unlike the proposed adapter, the existing means of organizing distributed computing require a significant modification of applications before using them in the cloud.

## III. The dataflow adapter model

We propose the following model of the dataflow adapter for integrating local applications into a distributed system (Fig. 1):

- The adapter is implemented and distributed as a separate executable file. It requires access to the executable file of a data processing application and a network connection.
- After starting the adapter requests a work configuration from the predefined server or database, connects to the Kafka cluster and instantiates producers and consumers with the supplied settings.
- The adapter runs the application and transmits through Kafka message about successful start or information about runtime errors. Adapters' work readiness is checked to begin streaming the source data only when all the applications are ready to accept it.
- Every data consumer and producer is paired with the corresponding input or output file of the application. While application executable is running, the adapter ensures data transfer between files and data queues of the streaming platform. Named pipes can be used instead of ordinary files, which permits to eliminate costs of hard disk operations.
- The adapter measures the speed and the latency of data processing as well as computes the duration of data

978-1-7281-0359-4/18/$31.00 ©2018 IEEE
DOI 10.1109/UCC-Companion.2018.00023

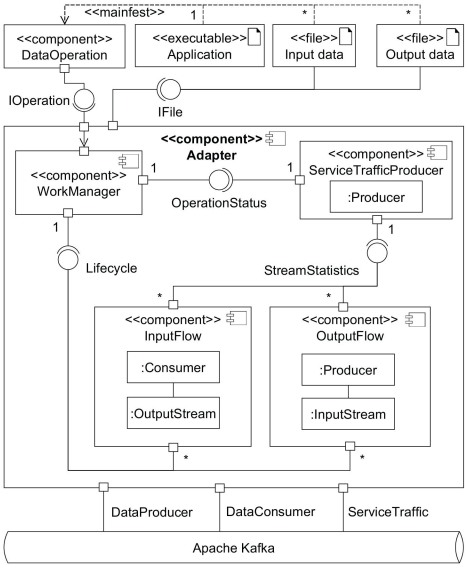


Fig. 1. The component diagram of the dataflow adapter

transfer between files and the streaming platform. Collected statistics are transferred through Apache Kafka as the separate data stream, which can be consumed by the modern data warehouses or displayed in real time by the graphical interface.

## IV. Implementation, Testing, and Future Work

To test the proposed model an experimental system is developed, consisting of the adapter, the coordinator, and the web interface. The coordinator is responsible for managing a set of adapters executed inside the Docker containers and collecting the statistics they transmit. Each adapter informs the coordinator of its container identifier and receives a work configuration, describing interaction with the application and Apache Kafka. The web interface is developed for editing adapter configurations, running data processing and viewing the statistics received from the coordinator via the WebSocket protocol. To develop the adapter and the coordinator, the Go language and librdkafka library is used; the web interface is implemented in TypeScript using React and Material Components.

The system was validated on the smart plugs recordings processing problem presented DEBS 2014 challenge [6]. The console application counting the median of provided recordings in real time was developed and tested in two series of experiments, during the first series processing the initial data file and during the second – a data stream transmitted by the adapter. The collected statistics provided in Table 1 show that the dataflow adapter work does not lead to significant performance degradation or significant delays in data transmission. Fig. 2 shows the collected statistics displayed in the web interface.

## V. Conclusions and Future Work

In this paper, we proposed the concept and the model of the adapter, which permits applications to handle Apache Kafka input and output data streams like data files. To test the proposed model, an experimental system was developed and validated using open DEBS IoT datasets. The statistics, that was gathered during the test show that the usage of the dataflow adapter work does not lead to performance degradation or significant delays in data transmission. In the future work, we plan to use the developed adapter for implementation of scalable data processing systems, that would be able to include legacy applications as computing services.

**Table 1.** Results of the experiments

| Metric | First series | Second series |
|---|---|---|
| Transfer speed | 530Kb/s | 470Kb/s |
| Average transfer rate | 11400msg/s | 10200msg/s |
| Average latency | 250ms | 390ms |
| Average work time | 12.5s | 13.1s |

## References


[1] A. Theorin, K. Bengtsson, J. Provost, M. Lieder, C. Johnsson, T. Lundholm, and B. Lennartson, "An event-driven manufacturing information system architecture for Industry 4.0," *Int. J. Prod. Res.*, vol. 55, no. 5, pp. 1297–1311, 2017.

[2] O. Carvalho, E. Roloff, and P. O. A. Navaux, "A Distributed Stream Processing based Architecture for IoT Smart Grids Monitoring," in *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing*, 2017, pp. 9–14.

[3] "Apache Kafka. A distributed streaming platform." [Online]. Available: https://kafka.apache.org. [Accessed: 29-Sep-2018].

[4] A. Weber, "GE 'predix' the future of manufacturing," *Assembly*, vol. 60, no. 3, pp. GE70-GE76, 2017.

[5] "Confluent: Apache Kafka as a Service, Streaming Platform for the Enterprise." [Online]. Available: https://www.confluent.io. [Accessed: 29-Sep-2018].

[6] Z. Jerzak and H. Ziekow, "The DEBS 2014 grand challenge," *Proc. 8th ACM Int. Conf. Distrib. Event-Based Syst. - DEBS '14*, 2014.


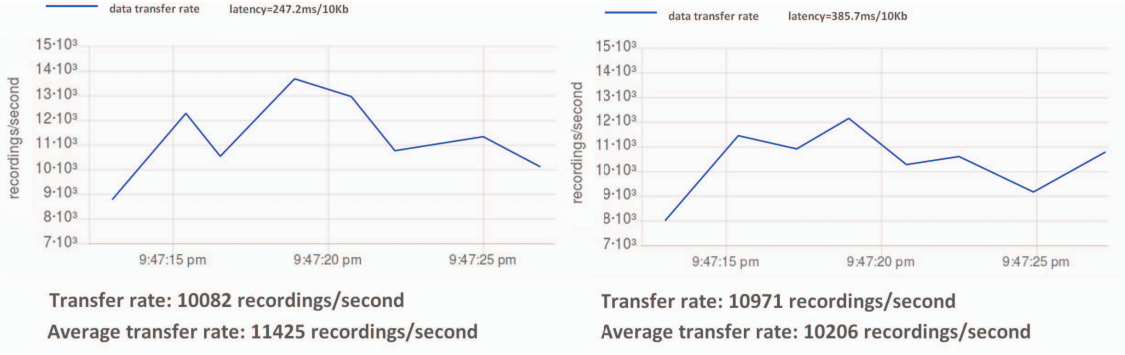


Fig. 2. Experiment statistics displayed in the web interface