Scientific Micro-Workflows: Where Event-Driven Approach Meets Workflows to Support Digital Twins

Ameer B. A. Alaasam[1], G. Radchenko[1(✉)], A. Tchernykh[1,2], K. Borodulin[1], A. Podkorytov[1]

[1] South Ural State University, Chelyabinsk, Russia, [2] CICESE Research Center, Ensenada, Mexico

Abstract – The Digital Twin is a hierarchical system of mathematical models and computational methods, which provides near real-time synchronization between the state of the real-world process and its virtual copy. It can be represented as a computational workflow, where the nodes are the computing services and other digital twins linked together by the data flow. The distinguishing characteristic of the digital twin is its capability to capture, transfer, and analyze real-time streaming data from Industrial Internet of Things equipment. Scientific workflow management systems could be used to design and deploy digital twins. But they have limited capabilities to integrate the streaming data flows and rely on tightly-coupled relations between computational services. In this paper, we propose an Event-Driven Approach to solve these limitations combining the power of scientific workflows, the flexibility of containers technology, and robustness of the distributed streaming approach. We also provide a reference implementation of this approach by developing new actors to extend Kepler scientific workflow management system capabilities to support data stream consumption and production operations using Kafka framework. This approach allows us to develop a workflow as a set of loosely coupled services (Micro-Workflows, inspired by Microservices) that deployed inside Docker containers and communicate through streaming middleware.

Keywords: Digital Twin · Microservice · Micro-Workflow · Scientific Workflow · Kafka · Kepler · Docker · Streaming.

1 Introduction

Over the last 5 years, we observe an exponential growth in the field of "Smart Production" concept [1]. Currently, we identify several global-level programs that are focused on its development, such as “Industry 4.0”, developed in Germany, “Smart Manufacturing Leadership Coalition” in the United States, “Digital Economy” program in the Russia, etc. All those programs use software and hardware systems to analyze data from several types of sensors by various types of models: mathematical, computational, data, etc. In the “Smart Industry”, a set of such models is called "Digital Twins" (DT) - virtual models, representing processes, systems and equipment.

According to Glaessgen and Stargel [2], DT is an integrated multiphysics, multiscale, probabilistic ultra-realistic simulation of an end-product or system that uses the best available physical models and sensor updates to mirror the life of its corresponding physical twin. The DT concept contains three main parts [3]: physical products in a real space; virtual products in a virtual space, and connections of data and information that ties the virtual and real products together.

DT uses data gathered from the sensory systems on production lines to predict failures of machinery, optimize the quality of the products, and reduce the ecological footprint from facilities. Gartner [4] predicts that by 2021, half of the large industrial companies will use DTs, resulting in gaining a 10% improvement in effectiveness.

DT applies mathematical models for simulation of the processes using methods such as data mining, finite element analysis, etc. [5], which define specific requirements for the computational resources. For example, data mining methods require a high volume of high-throughput storage resources, supporting the “Active Storage” paradigm [6]. Models that use finite element method would require high-performance computing systems (or supercomputers) [7].

DT can be described as a sequence of jobs that perform required functionality linked together by a set of edges that represent data dependencies between jobs. Such an approach can be implemented as a scientific workflow (SWF) [5], [8]. SWF is defined, managed, executed, and monitored by Scientific

Workflow Management Systems (SWfMS), such as DiVTB [9], Pegasus [10], Kepler [11], ASKALON [12], and tGSF [13], [14].

SWfMS can manage the automation of the required experimental scenario for the complex e-Science and data-driven systems (such as DTs). It makes scientists focus on their research and not on details of computation management. For example, Pegasus framework allows users to represent workflows at an abstract level, while it takes care of the particulars of the execution systems [15]. Similary, the main goal of the Kepler SWfMS is to support different execution scenarios [16].

In the streaming and cluster-computing frameworks such as Spark, Flink [17] and CA-DAG [18], [19], the context is different. They rely on fine tuning of execution parameters and in-depth understanding of the underlying architectural choices. The same challenges occur with the Google Cloud Dataflow [20], which is limited to the usage of the Google Cloud computational resources. In Kepler, a user can develop and use his own modules to manage different execution behaviors in different environments, including private computational resources [16].

On the other hand, the distinguishing characteristic of the DT is its capability to capture, transfer, and analyze real-time streaming data from Industrial Internet of Things (IIoT) equipment for tuning and actualization of their virtual state [21].

To implement this, we need to face two significant challenges: heterogeneity of instrumentation and complexity of data stream processing [22]. An ideal solution to these challenges is a streaming data middleware providing modularity, flexibility, and control over complex data interactions [23].

Currently, the vast majority of the SWFs supports the execution of legacy applications that are orchestrated as scripts operating on files [24]. The data streaming is not fully-featured in SWfMS. The only example of data streaming implementation in SWfMS is input data streaming and visualization implemented by Kepler [25]. Unfortunately, output data streaming feature is not supported.

SWfMS is formerly applied over a number of execution environments such as workstations, clusters/Grids, and supercomputers. But running one large-scale SWF in these environments faces a series of obstacles. For example, the limitations appear, when we deal with big data problems, including data scale and computation complexity, resource provisioning, and collaboration in heterogeneous environments [26].

The containerization approach arises as a potential solution to this problem [27]. Also, in SWF, tasks are tightly coupled with each other. These architectural features of SWF often conflict with the loosely coupled distributed applications such as extremely loosely coupled and highly distributed Event-Driven Architecture (EDA) approach [28].

In this paper, we propose an EDA to implement DT that combines the power of scientific workflows, the flexibility of containers technology, and robustness of the distributed streaming approach. We also provide a reference implementation of this approach extending Kepler SWFMs capabilities and developing new Kepler actors to support stream consuming and producing operations with modern Kafka streaming middleware.

2 Micro-Workflows Approach

SWfMS allows scientists to focus on their research by automating the required experimental scenario for the complex computational system. However, running one large-scale SWF with tightly-coupled dependencies, sequential execution, and limited streaming support does not meet the event-driven approach that is necessary for the digital twins. Also, it increases the complexity of deployment over different execution environments.

We propose a novel approach to redesign the large SWF into a set of smaller loosely-coupled Micro-Workflows (MWF) that act as independent computational services, which communicate with each other exchanging messages by means of the streaming middleware.

This approach allows feeding IoT low-latency data to the SWF as a data source. It also allows independent development, deployment and increases the potential of horizontal scaling. For example, MWFs provided low-latency response can be deployed near the IoT equipment.

By this way, every single MWF can be reused in various parts of the main workflow and executed several times in parallel. Also, each MWF can be deployed in a container.

The first step toward MWF is to redesign the original SWF into a set of SWFs linked together through the external data flow (see Fig. 1).

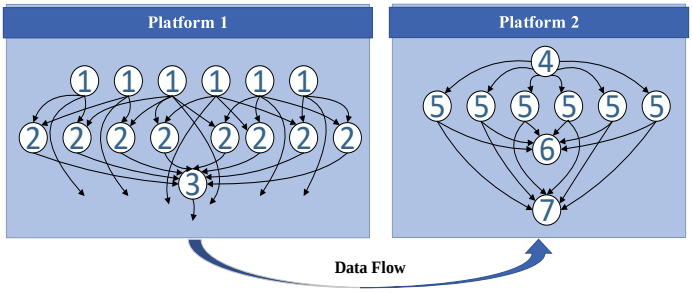

Fig. 1. Division of a single workflow into two Micro-Workflows

Each MWF must own a Consumer Node (CN) to consume data from the streaming environment and Producer Node (PN) to publish the results into streaming middleware. The PN node inputs are the outputs of the preceding nodes in the same MWF. PN packages the inputs as a message to be published into streaming middleware where being available for any consumer (see Fig. 2).

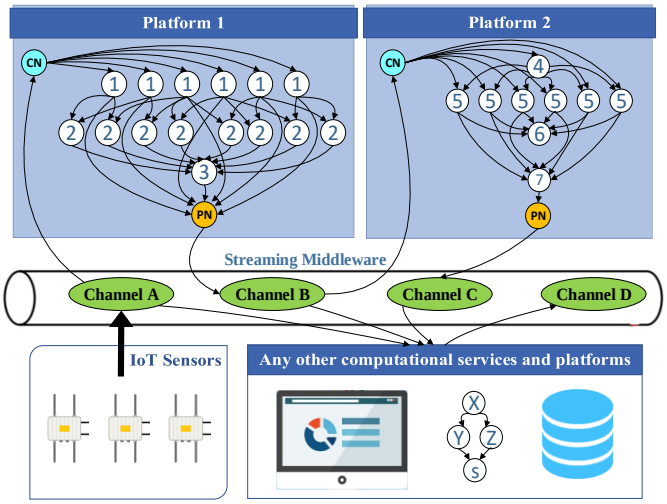

Fig. 2. Micro-Workflows concept using streaming middleware

3 Implementation and Testing

3.1 Data source

We use sensors data from DEBS[1] 2012 Grand Challenge: Manufacturing equipment. It includes a set of queries to process the data. The delay between two consecutive source data points is about 10 ms.

To test our approach, we consider the operators of the first query. Original data point includes 66 fields. Only seven of these fields (*s.bm05* till *s.bm10* and the timestamp *s.ts*) are used in the first query. The first set of operators (1 till 6) detects the change of state of input fields and emits them along with timestamps of the state change. The second set of operators (7 till 9) correlates the change of state of the sensor and change of state of the valve by calculating the time difference between the occurrence of the state changes and emit those along with timestamps to be as the following output data (*s58.dt, s58.ts, s69.dt, s69.ts, s710.dt, s710.ts*) (see Fig. 3). Note that this sequence requires aggregating each message or input with preceding message or input.

[1] http://debs.org/grand-challenges/

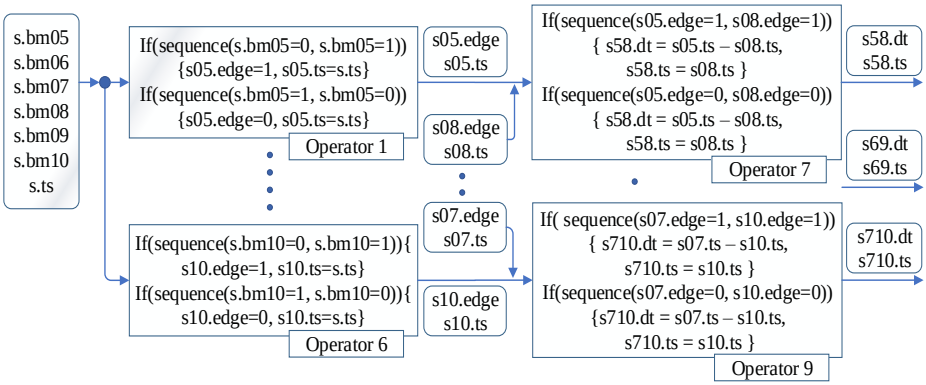

Fig. 3. The nine operators in the first DEBS 2012 Query

3.2 Testing the approach

We chose Kepler[2] as SWfMS, and Kafka[3] as a distributed streaming platform that supports horizontal scalability. When loosely coupled services interact with each other using messages, they have to agree on a generic message format called a schema. We use Confluent Platform[4] as the schema registry.

To test the efficiency of data exchange and processing, we pack our MWFs in Docker container and deploy them together with the streaming environment on a single computing node (Intel Core i5-2410M dual-core processor 2.3GHz with 8GB of RAM).

To provide integration testing, we develop custom actors for Kepler:

1- **KafkaConsumer** consumes messages from Kafka source, appends timestamp, aggregates each message with its predecessor, and sends them to DEBS2012_Op1to6 actor.

2- **DEBS2012_Op1to6** receives a pair of messages from KafkaConsumer, applies a series of operations, which reflect the operators 1 till 6, appends timestamp and sends the results to DEBS2012_Op7to9 actor.

3- **DEBS2012_Op7to9** receives input from DEBS2012_Op1to6 actor, aggregates each input with its predecessor, applies a series of operations, which reflect the operators 7 till 9, and sends a result to KafkaProducer actor.

4- **KafkaProducer** receives the input from DEBS2012_Op7to9, prepares it to be Avro message, appends timestamp and other parameters, and sends it to Kafka result topic.

Fig. 4 shows a screenshot of our experimental Kepler workflow.

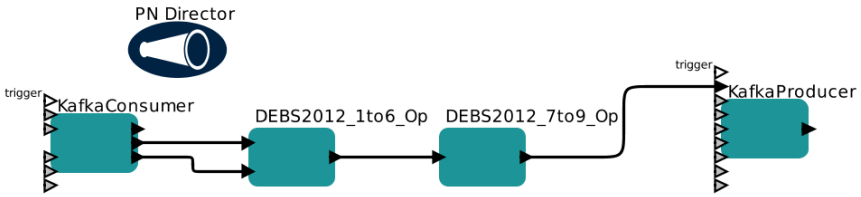

Fig. 4. Screenshot of our experimental Kepler workflow

3.3 Parameters and results

In addition to the original result fields required by DEBS 2012, we include the following fields to produced messages allowing to measure the efficiency of the proposed system:

1- **mX_orgts:** time, when the sensor sends the source message number X.

2- **mX_rcvts**: time, when Kepler receives the source message X from Kafka source topic.

[2] https://kepler-project.org/
[3] https://kafka.apache.org/
[4] https://www.confluent.io/

3- **K_sent_ts**: time, when KafkaProducer actor sends the result message to Kafka result topic.

To test the efficiency of data exchange and processing, we propose to measure the following characteristics:

1- **Av_SM** (average interval between source messages): the average time delay between two consecutive source data messages sent by the sensor to Kafka throughout the test.
2- **Av_RM** (average interval between result messages): the average time interval between two consecutive result messages produced by our MWF throughout the test.
3- **Av_TAT** (average turnaround time**)**: the average time interval required to produce a single result message by our MWF. Note that the interval starts from the time of receiving the third required source message (**m3_rcvts**) from Kafka source and ends at the time of sending the result message (**K_sent_ts**) to Kafka.

Results of our experiments are provided in Table 1. During one-hour test, our system processed more than 472 thousand messages with the average turnaround time about 1.38 ms.

Table 1. Testing results

Test time	1 hour
Number of messages	472279
Av_SM (millisecond)	7.62
Av_RM (millisecond)	7.62
Av_TAT (millisecond)	1.38

4 Conclusion

Scientific workflow management systems allow to design, execute and manage execution of workflows for various application scenarios. However, they face significant challenges such as tightly-coupled dependencies of jobs, sequential execution, and lack of data streaming support. The challenges become greater when the target system, such as a Digital Twin, demands event-driven processing. In this paper, we proposed a new architecture of Micro-Workflows that supports processing of streaming events from various sources (such as sensors) inside computational workflows. The implementation and testing of this architecture using Kepler SWfMS and Kafka streaming processing solution show that the proposed architecture provides a capability toward meeting the Digital Twins development processes with a reasonable overhead.

5 Acknowledgement

The work was supported by the The Russian Foundation for basic Research research project No. 18-07-01224 A and by Act 211 Government of the Russian Federation, contract № 02.A03.21.0011.

References

1. P. Korambath et al., “Deploying kepler workflows as services on a cloud infrastructure for smart manufacturing,” Procedia Comput. Sci., vol. 29, pp. 2254–2259, 2014.
2. E. H. Glaessgen and D. S. Stargel, “The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles,” in 53rd Structures, Structural Dynamics and Materials Conference - Special Session: Digital Twin, 2012, no. April, pp. 1–14.
3. M. Grieves, “Digital Twin: Manufacturing Excellence through Virtual Factory Replication,” 2014. [Online]. Available: http://innovate.fit.edu/plm/documents/doc_mgr/912/1411.0_Digital_Twin_White_Paper_Dr_Grieves.pdf.

4. C. Pettey, “Prepare for the Impact of Digital Twins,” 2017. [Online]. Available: https://www.gartner.com/smarterwithgartner/prepare-for-the-impact-of-digital-twins/.
5. P. Korambath et al., “A smart manufacturing use case: Furnace temperature balancing in steam methane reforming process via kepler workflows,” Procedia Comput. Sci., vol. 80, pp. 680–689, 2016.
6. Q. Xu, K. M. M. Aung, Y. Zhu, and K. L. Yong, “Building a large-scale object-based active storage platform for data analytics in the internet of things,” J. Supercomput., vol. 72, no. 7, pp. 2796–2814, 2016.
7. L. Wang, G. Wang, and C. A. Alexander, “Confluences among Big Data, Finite Element Analysis and High Performance Computing,” Am. J. Eng. Appl. Sci., vol. 8, no. 4, pp. 767–774, 2015.
8. S. Iturriaga, S. Nesmachnow, A. Tchernykh, and B. Dorronsoro, “Multiobjective Workflow Scheduling in a Federation of Heterogeneous Green-Powered Data Centers,” in 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2016, pp. 596–599.
9. G. Radchenko and E. Hudyakova, “A service-oriented approach of integration of computer-aided engineering systems in distributed computing environments,” in UNICORE Summit 2012, Proceedings, 2012, vol. 15, pp. 57–66.
10. E. Deelman et al., “Pegasus, a workflow management system for science automation,” Futur. Gener. Comput. Syst., vol. 46, pp. 17–35, May 2015.
11. I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock, “Kepler: an extensible system for design and execution of scientific workflows,” Sci. Stat. Database Manag. 2004. Proceedings. 16th Int. Conf., vol. I, pp. 423–424, 2004.
12. T. Fahringer et al., “ASKALON: a Grid application development and computing environment,” 6th IEEE/ACM Int. Work. Grid Comput. 2005., 2005.
13. A. Hirales-Carbajal, A. Tchernykh, T. Roblitz, and R. Yahyapour, “A Grid simulation framework to study advance scheduling strategies for complex workflow applications,” in 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010, pp. 1–8.
14. A. Hirales-Carbajal, A. Tchernykh, R. Yahyapour, J. L. González-García, T. Röblitz, and J. M. Ramírez-Alcaraz, “Multiple Workflow Scheduling Strategies with User Run Time Estimates on a Grid,” J. Grid Comput., vol. 10, no. 2, pp. 325–346, Jun. 2012.
15. E. Deelman et al., “Pegasus: a framework for mapping complex scientific workflows onto distributed systems,” Sci. Program., vol. 13, no. 3, pp. 219–237, 2005.
16. M. Płóciennik et al., “Approaches to distributed execution of scientific workflows in Kepler,” Fundam. Informaticae, vol. 128, no. 3, pp. 281–302, 2013.
17. O.-C. Marcu, A. Costan, G. Antoniu, and M. S. Perez-Hernandez, “Spark Versus Flink: Understanding Performance in Big Data Analytics Frameworks,” in 2016 IEEE International Conference on Cluster Computing (CLUSTER), 2016, pp. 433–442.
18. V. Miranda, A. Tchernykh, and D. Kliazovich, “Dynamic communication-aware scheduling with uncertainty of workflow applications in clouds,” in Communications in Computer and Information Science, 2016, vol. 595, pp. 169–187.
19. D. Kliazovich, J. E. Pecero, A. Tchernykh, P. Bouvry, S. U. Khan, and A. Y. Zomaya, “CA-DAG: Modeling Communication-Aware Applications for Scheduling in Cloud Computing,” J. Grid Comput., vol. 14, no. 1, pp. 23–39, Mar. 2016.
20. “Cloud Dataflow: Simplified stream and batch data processing, with equal reliability and expressiveness.” [Online]. Available: https://cloud.google.com/dataflow/. [Accessed: 30-May-2018].
21. K. Borodulin, G. Radchenko, A. Shestakov, L. Sokolinsky, A. Tchernykh, and R. Prodan, “Towards Digital Twins Cloud Platform : Microservices and Computational Workflows to Rule a Smart Factory,” Proc. the10th Int. Conf. Util. Cloud Comput. - UCC ’17, no. December, pp. 209–210, 2017.
22. T. Fountain, S. Tilak, P. Shin, P. Hubbard, and L. Freudinger, “The Open Source DataTurbine Initiative: Streaming Data Middleware for Environmental Observing Systems,” in 33rd International Symposim on Remote Sensing of Environment; 4-9 May 2009; Stresa; Italy, 2009.

23. S. Tilak, P. Hubbard, M. Miller, and T. Fountain, “The Ring Buffer Network Bus (RBNB) DataTurbine Streaming Data Middleware for Environmental Observing Systems,” in Third IEEE International Conference on e-Science and Grid Computing (e-Science 2007), 2007, pp. xii–xiii.
24. D. Zinn et al., “Towards Reliable, Performant Workflows for Streaming-Applications on Cloud Platforms,” in 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2011, pp. 235–244.
25. R. M. Badia, E. Ayguade, and J. Labarta, “Workflows for Science : a Challenge when Facing the Convergence of HPC and Big Data,” SuperFri.org, vol. 4, no. 1, pp. 27–47, 2017.
26. Y. Zhao, Y. Li, I. Raicu, and C. Lin, “Migrating Scientific Workflow Management Systems from the Grid to the Cloud,” Cloud Comput. Data-Intensive Appl. Springer, New York, NY, 2014.
27. R. Qasha, J. Cała, and P. Watson, “Dynamic Deployment of Scientific Workflows in the Cloud using Container Virtualization,” Proc. - 2016 8th IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom 2016, pp. 269–276, 2016.
28. A. Theorin et al., “An event-driven manufacturing information system architecture for Industry 4.0,” Int. J. Prod. Res., vol. 55, no. 5, pp. 1297–1311, 2017.