Streaming information integration unit for mass customization

Liu Yang¹, Xiangqian Ding¹, Ruichun Hou¹*, Ye Tao² and Ning Yang¹
¹College of Information Science and Engineering, Ocean University of China, Qingdao, China
²College of Information Science and Technology, Qingdao University of Science and Technology, Qingdao, China

*Corresponding author e-mail: houruichun@ouc.edu.cn

Abstract. With the development of the information industry, there are a large number of software systems in enterprises. Due to the lack of top-level design, the data assets in these systems have formed data silos. Under the trend of mass customization production, it is of great significance to combine factory manufacturing data with user data to connect traditional information silos into user-centric intelligent production lines. Traditional offline batch processing tools cannot real-time and automated decision-making in mass customization. Based on the streaming programming model and distributed computing technology, this paper proposes a streaming information integration unit for mass-customized industrial Internet platforms. This unit provides information sharing capabilities in the process of enterprise information management, eliminates information blind spots in the service and workshop layers, and integrates internal information resources of the enterprise. Through the realization of two-way real-time transmission of production information, the user experience is improved and the delivery cycle of mass customization is shortened. In addition, data acquisition tests are performed using factory line temperature sensor data. The experimental results show that compared with the traditional batch processing scheme, the acquisition delay of this scheme is much lower than that of traditional batch processing tools, which provides a feasible solution for information integration in mass customization systems.

1. Introduction

Driven by the Internet of Things, big data, artificial intelligence and other technologies, the current manufacturing industry is developing towards service and collaboration. Mass customization has become the mainstream mode of consumer manufacturing in the future. Because a large number of users choose personalized products, users participate in product design and product manufacturing processes, which requires high real-time collection of manufacturing information and service information. At the same time, customized products are highly personalized, with a variety of structures and assembly technology, it is bound to increase the difficulty of manufacturing, leading to the reduction of production efficiency. Therefore, enterprises need manufacturing process informatization and digitalization[1, 2].

For historical reasons, data chimneys, data silos, and fragmented applications are widely used in enterprises. At present, most enterprises use ERP for enterprise resource management. However, at the
user service layer and workshop manufacturing level, the ERP system cannot track the user's behavior and status information in the manufacturing process in real time. The user cannot participate in the whole process[3]. The current system cannot provide real-time two-way feedback for user's personalized customization needs change and status during production process. There are blind spots in information among users, enterprises, factories, and logistics. This results in reduced flexibility during production.

There are many commercial and open source information integration tools, but most of them are stand-alone architectures or off-line batch models[4]. The use of large-scale sensors in the current plant generates a large amount of heterogeneous data. In order to make efficient use of these data, information integration unit is used to collect and process sensor data in real time. Respond to events as they occur and provide insight[5]. Therefore, this paper mainly solves two problems. First, introduction of information integration unit to improve the freedom of data flow at each layer in the mass customization system. Second, design a high-throughput, low-latency streaming information integration unit for mass customization production, based on the streaming programming model and distributed computing technology.

2. Related work

2.1. Streaming data processing architecture

In mass customization systems, information integration relies on systems such as ERP, MES, monitoring, and user services. The production status collection, data processing and analysis of the workshop need real-time interaction and management to make the production system have dynamic response capabilities. Streaming data processing technology is a popular research direction in the field of big data. The sensor continuously generates data, and the data is streamed into the working nodes in the memory, which is suitable for areas with high real-time requirements. The modules of the streaming processing system mainly include data collection, message queue, data conversion, data aggregation and output[6].

There are two main architectures for implementing a streaming data processing system. One is the Lambda[7] architecture, which supports both offline and real-time computing scenarios. The main idea is to divide the big data system architecture into multiple levels. Batch processing layer, real-time processing layer, and service layer. The batch processing layer is responsible for data set storage and pre-calculation of the full data set, and the real-time processing layer is mainly responsible for calculating incremental data. The service layer is used to combine the results of the batch and real-time processing layers and return them to the user. In the Lambda architecture, an algorithm needs to be implemented twice. Once for a batch processing system and once for a real-time processing system. Both algorithms need to produce consistent results, as shown in Fig. 1.

![Lambda architecture data flow](image)

**Figure 1.** Lambda architecture data flow.

The other is the Kappa architecture, which removes the redundant part in the Lambda architecture and joins the message queue to replay data to achieve historical data statistics. As shown in Fig. 2.
Using a unified calculation engine to process the data, the user only needs to implement the code once, reducing the development work of the programmer[8-10].

![Figure 2. Kappa architecture data flow.](image)

### 2.2. Industrial Internet information integration

The collection and fusion of information is an important data-driven support component in mass customization systems. This unit extracts information flow from ERP, MES, SCM and other systems. Establish data associations based on business, and share, convert, and generate business statistics among systems[11]. At present, the more popular tools for industrial and agricultural Internet system information integration technology[12] can be divided into two types. Data processing products developed by database vendors, such as IBM infosphere Information Server. Open source data processing tools, such as Sqoop, Kettle, etc. These tools have specific processes for data processing, and their data sources are often relatively single. There are limitations to the integration of complex heterogeneous data from multiple data sources. For example, Sqoop uses the MapReduce programming model and uses YARN for resource scheduling. It belongs to an offline batch processing architecture and cannot support business scenarios with high real-time requirements. In order to reduce the delay of information integration, a real-time information flow is added to the offline architecture. The Lambda architecture is introduced to ensure the consistency of data by converging the calculation results downstream. This architecture requires two programming implementations[7], and later maintenance is more difficult.

In summary, an information integration unit that supports multi-source heterogeneous information integration, high performance, low latency, and easy to maintain is needed in a mass customization system.

### 3. Design and implementation

#### 3.1. Architecture Design

According to the data heterogeneity and real-time requirements of mass customization systems, the design ideas of information integration units for mass customization systems proposed in this paper mainly include the following points.

(a) Adopt plugin IO interface to realize multi-source heterogeneous information collection and loading. Customize the reader / writer by implementing the interface to increase the flexibility of the data source. After reading the data, Reader will encapsulate the original data block and write it to the message queue after using the filter to convert it.

(b) Using the low latency and flexibility of the flow computing programming model to implement Kappa architecture's batch-stream integrated information integration tool to process and transform heterogeneous data from multiple sources and aggregate the data to provide insight.

(c) Implement data processing tools in a distributed environment, based on the distributed asynchronous snapshot algorithm Chandy-Lamport[13] to ensure fault tolerance of applications. Adopt container virtualization technology to provide elastic computing resources for applications.
Combined with the above factors, the system includes information acquisition module, flow processing module, information loading module, configuration center module and global monitoring module, as shown in Fig. 3.

![Diagram of Information Integration Unit Architecture](image)

**Figure 3.** Information integration unit architecture.

The information acquisition module includes two parts: a plug-in IO interface and a message queue. The plug-in IO interface is mainly responsible for collecting data from various production systems. The plug-in IO interface can improve the flexibility of the system. Users can freely choose different data source types and integrate them into this module by implementing interfaces. Due to the variety of data sources in production systems, the data formats vary widely, and inconsistent formats are widespread. While the data is being collected, the data can be preliminarily cleaned and filtered. After the data format is unified for a specific data source, the data is encapsulated. Add data source information to analyze data kinship, and then write to the message queue cluster.

![Diagram of Heterogeneous Information Flow Integration](image)

**Figure 4.** Heterogeneous information flow integration.

Stream processing is the core module of this unit. As shown in Fig. 4, the production process information and user service information are fully integrated through a continuous information flow. Obtain the basic data of material, purchase and sales from the ERP system. It is integrated with peripheral system data such as material demand, cost data, operation process and equipment management in MES system. Real-time statistics of the production order list of process ingredients, parts warehousing information, material cost, labor cost and other customized indicators. Real-time feedback and early warning according to the threshold value issued by the configuration center.
The programming model, DAG\cite{14} generation, and distributed asynchronous snapshot algorithm are the three components of this module. The programming model defines the standard architecture of streaming computing, data transformation operation, aggregation operation and trigger operation conditions. The data conversion logic written by the user is encapsulated into a running graph using a DAG generation module. Optimize the running graph according to the specified degree of parallelism to generate a distributed running graph and submit it to the container cluster. The distributed asynchronous snapshot algorithm is responsible for fault tolerant processing and establishes a global state snapshot while the distributed application is running. Use the state snapshot to restore the runtime state when the program fails.

After the data is processed by the above modules, it will be output to the message queue cluster. The information loading module is responsible for obtaining information flow from the message queue cluster. Output to factory internal system or peripheral service system according to custom output logic.

3.2. Implementation of information acquisition / loading module

As mentioned above, the information acquisition module supports data extraction and loading of multiple heterogeneous data sources through plug-in Reader / Writer. Due to the diversity of data storage systems in the mass customization system, this scheme provides an abstract data reading interface to support any data source type. Users only need to implement specific data operations based on specific data sources. There are several steps to implementing Reader / Writer.

Step 1. Init method is responsible for the initialization of Reader. Pull configuration information from configuration center, splicing of data source connection information, etc.

Step 2. Establish data source connection according to configuration information. Prepare to get data from the data source.

Step 3. Invoke method is responsible for data acquisition / loading, extracting data from the data source and exporting it to the cache.

Step 4. When the system stops reading / writing data, it will first call the pause method. In the pause method, the cached data will be cleared and output to the message queue cluster.

Step 5. Close the data source connection.

The open method in the Writer plug-in is responsible for establishing two connections, the message queue cluster connection and the external data system connection. Also in the invoke method, the system gets the data from the message queue cluster and outputs it to the external data system for consumption by other programs.

![Figure 5. Data block encapsulation structure.](image)

The Reader plugin contains a memory buffer that triggers a data encapsulation operation when the data in the cache reaches a threshold. The threshold can be set through the configuration center. Get data blocks from multiple data sources and add data headers to identify the kinship of the data. The data header includes job ID, data source ID, data extraction time, and metadata information. The
structure of the encapsulated multivariate data block is shown in Fig. 5. The system provides a pluggable data filter when data enters the buffer. Users can choose to turn on or off data filters based on business requirements. Filter for data preliminary cleaning, conversion of different data formats and other custom operations. The filter can be turned off when running a service with high real-time requirements. Filters can be turned on when heterogeneous data types are complex and need to be cast.

3.3. Implementation of Streaming process module

DataFlow[15] proposes a model that supports both batch and stream processing. The core capability is the ability to provide sequential processing based on EventTime for out-of-order streaming data and window aggregation based on the characteristics of the data itself. And balance the relationship between correctness, delay, and cost. DataFlow model considers batch processing to perform calculations on a limited data set, and stream processing to perform calculations on an unbounded data set. his module pulls data from the message queue cluster, and the data flows into the distributed system for processing operations. Data processing is divided into data conversion and data aggregation. Developers can customize the data conversion logic by implementing the methods in the interface. The data aggregation operation partitions the data according to the fields specified by the user. Data with the same specified fields will be divided into the same partition.

Data processing includes multiple subtasks, and there are dependencies between tasks. After the previous task processes the data, the subsequent tasks can continue to run. he dependencies between tasks are represented by a directed acyclic execution graph (DAG). By analyzing the data processing program, the system generates a DAG to describing the working topology, which is then optimized based on the specified parameters and submitted to the distributed system for computation. he main function of DAG is to describe the working topology of streaming computation, as shown in Fig. 6. Its node is an operator in a distributed program, which is responsible for the operation of data conversion and aggregation, and the edge represents the flow direction of data. et the information of a variety of data sources from the Reader, and load the calculated results into the data bus through the conversion, aggregation and other operations of multiple operators.

![Figure 6. Directed acyclic execution graph.](image)

In distributed computing, it is inevitable that a node goes down or the program fails. In order to ensure the correctness of the calculation results, Chandy-Lamport algorithm simplifies the distributed system into a finite number of processes and channels between processes. Processes run in different containers. The goal of the distributed snapshot algorithm is to record the state of these processes and messages in the channel. The main process is Initializing a snapshot, which is initiated by any process in the system. Propagation a snapshot, other processes in the system start to create snapshots one by one. Terminating a snapshot ends the creation of a snapshot.

4. Experiments

To verify the performance of the information integration unit. Compare this solution with a traditional batch acquisition tool. Perform data insertion performance tests of different orders of magnitude. The data batch processing tool Sqoop and this solution are deployed on servers with the same performance. The machine configuration is: Intel (R) Xeon (R) Silver CPU @ 1.80 GHz (2 processors), 256GB memory, 4TB disk. Operating system are Centos7.0. Cloud RDBMS is MySQL 5.7, Sqoop version is 1.4.7, and the HBase version is 1.4.12. The implementation steps are as follows.

1. Store the collected temperature sensor data into a cloud RDBMS.
2. Extract 4 sets of data from the cloud RDBMS. The data volume is 1,000, 10,000, 100,000, and 1,000,000.
3. Use the batch data processing tool Sqoop to read the data in the RDBMS and write it to HBase. Record the time from the application startup to the end of the insert. Repeat 3 times to obtain the average value.
4. Use this solution to read the data in the RDBMS and write it to HBase, record the time from the application startup to the end of the insert, repeat 3 times to get the average value.

![Figure 7. Data Insertion Performance Comparison.](image)

Insert performance data as shown in Fig. 7. The results show that when the amount of data is large, there is not much difference between the batch data collection tool and this solution, and the use of streaming processing has only a slight advantage. However, in terms of small-scale data acquisition, the writing delay of this scheme is significantly better than batch processing. In the mass customization system architecture, this solution is used for real-time heterogeneous data collection and information sharing. Compared with traditional batch collection tools, this solution has better collection performance and lower time delay. When data volume below 100K, the insertion time required by this solution is only 14% of Sqoop, and it is generally better than traditional batch information collection tools.

5. Conclusion
This article introduced information integration schemes in current mass customization systems. Aiming at the lack of real-time and multi-source performance of existing tools, this paper proposed a stream data processing solution supporting multi-source heterogeneous information integration. Based on the plug-in IO interface, streaming programming model and distributed computing technology, the system achieves high performance and low delay information integration unit. This solution can help enterprises eliminate information blind areas at all levels in the process of information management. The data insertion performance of the traditional batch data processing method and this scheme is tested and compared with the data of the factory pipeline temperature sensor. Experimental results show that this solution is efficient in collecting large amount of data and real-time streaming data. Especially in the small-scale data acquisition experiment has outstanding performance. It can satisfy the real-time integration and automatic decision requirements of mass data in mass customization system, and has important reference significance to the construction of enterprise mass customization system.
6. Acknowledgments

This work was financially supported by the 2018 National Key Research and Development Program Project No. 2018YFB1702902, 2018 Qingdao Science and Technology Program Project No. 18-1-1-98-zhz fund.

References

[1] R. Y. Zhong, C. Xu, C. Chen, and G. Q. Huang, Big data analytics for physical internet-based intelligent manufacturing shop floors, International journal of production research, vol. 55, no. 9, 2017, pp. 2610-2621.

[2] C. Ji, S. Liu, C. Yang, L. Wu, and L. Pan, Ibdp: An industrial big data ingestion and analysis platform and case studies, in 2015 International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI), 2015: IEEE, pp. 223-228.

[3] Y. Wang, H.-S. Ma, J.-H. Yang, and K.-S. Wang, Industry 4.0: a way from mass customization to mass personalization production, Advances in Manufacturing, vol. 5, no. 4, 2017, pp. 311-320.

[4] V. K. Singh, M. Taram, V. Agrawal, and B. S. Baghel, A Literature Review on Hadoop Ecosystem and Various Techniques of Big Data Optimization, in Advances in Data and Information Sciences: Springer, 2018, pp. 231-240.

[5] E. F. Ted Dunning, Streaming Architecture. O'Reilly Media, Inc., 2016.

[6] S. C. T. L. Akidau, Reuven Lax, Streaming Systems. O'reilly media, 2017.

[7] F. Cerezo, C. E. Cuesta, J. C. Moreno-Herranz, and B. Vela, Deconstructing the Lambda Architecture: An Experience Report, in 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), 2019: IEEE, pp. 196-201.

[8] M. Feick, N. Kleer, and M. Kohn, Fundamentals of Real-Time Data Processing Architectures Lambda and Kappa, SKILL 2018-Studierendenkonferenz Informatik, 2018.

[9] K. Kato, A. Takefusa, H. Nakada, and M. Oguchi, Construction Scheme of a Scalable Distributed Stream Processing Infrastructure Using Ray and Apache Kafka, Proceedings of 34th International Confer, vol. 58, 2019, pp. 368-377.

[10] N. Narkhede, G. Shapira, and T. Palino, Kafka: the definitive guide: real-time data and stream processing at scale. O'Reilly Media, Inc., 2017.

[11] F. Tao, Q. Qi, A. Liu, and A. Kusiak, Data-driven smart manufacturing, Journal of Manufacturing Systems, vol. 48, 2018, pp. 157-169.

[12] Y. Bi, P. Wang, Z. Wang, and S. Cheng, The architecture design for big data application system in apple industry, in Journal of Physics: Conference Series, 2019, vol. 1168, no. 3: IOP Publishing, p. 032075.

[13] K. M. Chandy and L. Lamport, Distributed snapshots: Determining global states of distributed systems, ACM Transactions on Computer Systems (TOCS), vol. 3, no. 1, pp. 63-75, 1985.

[14] D. Wu, S. Sakr, and L. Zhu, Big data programming models, in Handbook of Big Data Technologies: Springer, 2017, pp. 31-63.

[15] T. Akidau et al., The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing, Proceedings of the VLDB Endowment, vol. 8, no. 12, 2015, pp. 1792-1803.