The design and implementation of multi-source application middleware based on service bus

Yichun Li, Ningkang Jiang*
East China Normal University
No. 3663 North Zhongshan Road, Putuo, Shanghai, China
MOE International Joint Lab of Trustworthy Software
International Research Center of Trustworthy Software
*nkjiang@sei.ecnu.edu.cn

Abstract. With the rapid development of the Internet of Things(IoT), the real-time monitoring data are increasing with different types and large amounts. Aiming at taking full advantages of the data, we designed and implemented an application middleware, which not only supports the three-layer architecture of IoT information system but also enables the flexible configuration of multiple resources access and other accessional modules. The middleware platform shows the characteristics of lightness, security, AoP (aspect-oriented programming), distribution and real-time, which can let application developers construct the information processing systems on related areas in a short period. It focuses not limited to these functions: pre-processing of data format, the definition of data entity, the callings and handlings of distributed service and massive data process. The result of experiment shows that the performance of middleware is more excellent than some message queue construction to some degree and its throughput grows better as the number of distributed nodes increases while the code is not complex. Currently, the middleware is applied to the system of Shanghai Pudong environmental protection agency and achieved a great success.

1 Introduction
In recent years, the concept and application of Cyber-Physical Systems(CPS[1]) become popular quickly. The Internet of Thing(IoT) is called the third wave of the information industry after the computer and the Internet, which has rich research value and broad application market. In the time of IoT, almost all the physical devices are able to access networks, which is different from traditional Internet. According to the prediction of Forrester, the industrial value brought by IoT is 30 times larger than that brought by Internet and IoT will become the next trillion level information industry business. As the various data produced by all kinds of intelligent equipment, sensors and drivers, the whole field is facing the big challenge that how to process these multifarious formatted data when the scale is large and the businesses demand immediate feedbacks.

According to the opinion of Atzori et. al. [2], we can construct and implement the IoT application from three different dimensions: middleware, sensor and knowledge base. The specific components [3] are as follows.
The first component is wireless sensor networks infrastructure. It contains sensor, computing processing unit, data transmission unit and energy supply unit. The more complicated the sensor node is, the more channels it can cross[7]. The second component is the wireless sensor networks protocol stack. The node must be deployed in a self-organizing way. After deploying the protocol stack on a centre node, this network then has the ability to communicate with external [5]. One significant factor that affect the transmission efficiency is the network topology. The third component is application middleware. In order to provide convenience for the developers to achieve the demand according to the data collected by physical sensors, the application middleware connect the data obtained by network infrastructure with the application [6].

The organization of this paper is divided into seven sections. The second section is the related work about IoT and Cyber-physical System, including the basic concepts and characteristics. The third section introduces the overall architecture of the proposed application middleware. The fourth section elaborates the internal architecture of the proposed application middleware. The fifth section is about the optimized mechanisms of the proposed application middleware. The sixth section is about the application in practice. The sixth section shows the experiment result and practical applications of the middleware. The final section concludes the design and the implementation.

2 Related Works

2.1 Cyber-Physical System
The concept of IoT was first proposed in 1999. It is a kind of network that connects everything with the Internet by radio frequency identification (RFID), infrared sensors, GPS, laser scanners, gas sensors and other information sensing equipment to exchange information and communicate with each other, which makes an integration of intelligent identification, location, tracking, monitoring and management. Briefly, IoT is called “objects-connected internet”.

As the unity of computing processes and physical processes, Cyber-Physical System(CPS) is the next generation intelligent system which integrates computing, communication and controlling functions. There are three main characteristics. First, it interacts with physical process by human-machine interface and operates a physical entity in a remote, reliable, real-time, safe and collaborative way based on networked spatial structure. Second, it contains omnipresent environment perception, embedded computing, network communication and network control system engineering, which enables the functions includes computing, communication, precise control, remote collaboration and autonomy. Third, it extends new functions by full integration of computing and physical processes as well as the real-time communication between them according to the interact influences, responses and loops computed.

2.2 IoT application
Although the concept of IoT has lodged itself in the public mind and the basic sensor technology begins to take shape as well, there is lack of a typical example. The forms of the majority of applications are still remaining in the time of Internet. Thinking in Internet may be justifiable, but it is unable to satisfy the special demand in the time of IoT exactly.
Through the continuous exploration to the IoT application by the researchers, the basic framework eventually formed. The architecture of this framework is shown in Fig.1. It consists of three layers, including sensor layer, network layer and application service layer. In the sensor layer, all kinds of intelligent equipment such as wireless sensor, which has the ability to sense the physical entities and collect data in real time are in it. The network layer contains plentiful information networks. When the intelligent equipment accesses the network, it sends the data to the processor at once. After receiving the data from network layer, the service layer must transform the data into the object that can be managed. Thus, the service layer should provide the client with the well-defined interfaces. In this three-layer architecture, the application service layer plays the most important role. The traditional idea states that every system has its independent service layer. In this paper, we suggest that the service layer should be able to share data and ensure that several systems to run on the share data at the same time. Therefore, the service layer should meet the satisfaction as follows:

1. Since the service layer contains all the service provided for the client and is able to receive data from mobile equipment, it must play the role as both the data importer and the service provider, which means that the data is regarded as a resource and is one of the most significant characteristics in Web2.0.
2. It should be able to support the dynamic variable data entity.
3. A basic information process system must have the data processing module, workflow module and dispatch module. Each module should support hot-plugging.
4. There is not supposed to be interact influence between modules but they still need collaboration and data sharing.

According to the demand above, we design and implement the multiple data source application middleware based on service bus, which raises the convenience, flexibility, extendibility.

### 3 Overall Architecture

The enterprise-level application middleware platform developed in this paper provides the basic platform for various applications based on compatibility with underlying support. The developers can quickly deploy different types of data resources, implementation the business logic and the shows of application result according to his business. This platform uses a distributed architecture, aiming at providing the ability of processing big data. For the different data requests, the platform uniformly adopts the asynchronous messaging method to receive and process and exposing the internal interface to the outside through rest api...
mode. The underlying container extends the idea of IoC (Inversion of Control) [8-9] mode to debase the workload of developers and give more jobs to the platform, which reduce the probability of program error and the cost of maintain.

This application middleware platform adopts the low coupling design pattern based on service bus, combined with shared distributed message pool and basic service architecture technologies, which construct an enterprise application middleware platform that suitable for several application situations. According to the hierarchical structure of the system, the system is mainly consisting of following components showed in the Fig.2.

3.1 Application Container
The application container is based on Java Container and adopts IoC mode to combine the different business object with tool object. It is the basic component of all the container object lifecycle. During the start-up of the container, it can generate object instances of different businesses and tool objects as well as dynamic loading and unloading objects. The container also manages the lifecycle of modules, including service bus module, message process module and tool module. The service bus module is used to process the messages on the bus or send event messages to the bus. The message processing module is used to dispatch and translate bus messages and adapt other systems. The tool module is used to log and provide other functions has no relationship with business. At the same time, the container is responsible to connect the modules with bus object, which enables the communication of each function module through service bus.

3.2 Application Container Cluster
Several application containers are able to form an application container cluster. Through group setting by administrator, the service buses, configurations and shared storages of several servers in one LAN network will be connected and become a logic application server cluster. Application cluster allows different physical server members to adopt self-application configuration. For example, server A configures the web service and server B configures the response service Bml. As long as the interacting between the web service and the Bml service uses the bus interfaces opened by system, it can achieve the distributed response service across the physical servers.

Fig.2 The architecture of Application Middleware platform
3.3 Service Bus
Web service module receives the requests from the client, then translates them into bus messages and send them through the bus. A service bus module is able to process the requests from each module. As shown in the Fig.2, the web service module A receives a web service request (1), it translates the request into the broadcast message C and sends to the bus (2). Then, the module that can process the message C will process C (3) according to the requirement in the message and return the result to module A (4,5). Finally, module A returns the computed result to original outside requester (6).

4 Internal Architecture
In the procedure of designing the internal architecture of container middleware, we deploy the basic modules in the container and let them register their service through service bus to provide system level support for other applications. The internal architecture is shown in Fig.3.

4.1 Container Management
We can see that the management tasks of the container are achieved by the core framework of Spring, which is constructed based on the application context of Spring. It takes full advantages of Spring and reduces the workload of development as well as making a much lighter and clearer framework. The various services deployed on the top layer of container consists of one or more modules. The container provides the runtime environment, infrastructure and life cycle management and the modules provide the specific functions. By customizing different function modules, the developers are able to extend application by themselves to satisfy the different demands, which makes the platform have high compatibility.

4.2 Basic Module
The core of basic module is the service bus module, which can cross several containers and provides a series of interfaces. The other services can implement these interfaces to register the service bus module and have a uniform management schedule.

4.3 Various Services
System level service, register service and customizing service are the different modules that provide different services. They all implement the service bus interfaces and the external application can invoke these services through service bus. In the sending and result transition period of service request, we need uniformly formatted data.

5 Platform Mechanism
In order to adapt the application middleware platform to different practice systems better, we proposed three new mechanisms, which enhance the abilities of platform in high reliability, high portability, easy maintenance, extendibility and standardization.
Fig.3 The internal architecture of Application Middleware

5.1 Publish-Subscribe based on Service Bus
In the procedure of software development, the developers are always trying their best to avoid the over coupling between modules. MVC is one of the traditional design patterns aimed at reducing the over coupling between the different functions and keeping each module clean and independent. Another useful way is programming by AOP, which extracts the aspects in the process of business and reduce the coupling between the logic of different businesses. In this paper, the platform adopts the publish-subscribe mode based on Service Bus. The service bus covers all the containers and each container can access the service bus easily. Thus, the container can invoke all the services registered in service bus through the asynchronous messaging mode, although these services are provided by containers of other physical servers. All the modules are not only able to publish certain types of requests but also subscribe the messages interested in.

5.2 Bulk Distributed Data Process
In most cases, data processing is a very important function. However, the developers tend to focus more on the specific business flow rather than the definition and management of data entity. Thus, it is definitely necessary and meaningful to support the processing of underlying data. As the system works, the amount of data on it will accumulates and become larger, which will affect the overall performance. In order to ensure the speed of processing, we suggest to use distributed mode, which spreads the data and computing works across several machines to save the overhead of time.

In the distributed environment, we first read the input from master node and divide it into several small-scale sections. Then we dispatch these sections to the worker nodes and parallel launch the map operation. The worker nodes process the data iteratively to form a hierarchical tree structure. When the processing ends, the result will be divided into R partitions. Each partition corresponds to a reduce operation, then the master will notify the workers that the reduce works are allocated to with the location of the partition they are responsible for. Next, the reduce workers read and sort the middle key-value pair data they are responsible for, which makes the key-value pairs with the same key gathered. Since the different keys probably map to the same partition, the sorting is necessary. The reduce workers iterate the sorted middle key-value pairs and transfer the value corresponds to each unique key to the reduce function. The output of reduce function will be appended to the output file of this partition. Eventually, the master node merges these results according to the customized merge algorithm and output the final result.

In the procedure of performance optimization, there are two significant things. One is the definition of partition function that makes the data evenly distributed to each machine and the total amount of data transmission least at the same time. Another is the shuffle function, which is the middle operation between the map and reduce functions. The main function of shuffle is saving the data that cached in the map procedure to disks in order to provide convenience for reading of reduce. In this period, we try our best to save the data to different nodes evenly and reduce the cost of bandwidth consumption at the same time.

5.3 Dynamic Data Entity Type
The principal of the data process is a procedure during which the data is saved to the database and retrieved from the database to display the result. The traditional approach is to define the structured data entity in advance, manage the unstructured data by relation database and construct API using these data entities as well as operating the database by sql languages.

In practice, however, the data entity will change. Under this circumstance, the data format defined in advance is not able to use. To solve this problem, we have to modify the code of the running system. In the context of IoT, the application system collects data from physical world directly and the complicated changes of data can be predicted. In the application middleware platform proposed by this paper, we don’t
compulsorily map the data to the defined data entity any more, neither using the relation database to manage the metadata information. We separate physical data from relational databases successfully.

6 Applications

We developed the Shanghai Pudong environmental protection agency system based on the middleware platform proposed in this paper. This system monitors the contamination of thousands of industrials in Shanghai through wireless sensors. By now, the system is working properly.

The industrials are the main sources of environment pollution, but the traditional approach can't monitor the real-time contamination of industrials well. Through real-time monitoring of a variety of high-performance micro intelligent sensor, the data obtained will be sent through the wireless network and collected to the central server for processing. Finally, the result of processing will determine whether the discharge of sewage conforms to the national standards. Such an environmental detection system is a typical application of IoT, which is the connection between the information of the physical world and the Interne. The application middleware platform in this paper can provide good support for such a system. The whole architecture of the system is shown in Fig. 4.

To analyse the environmental monitoring business, we can see that the data is very complex and always in a dynamic change. Thus, we should use the variable and inflexible data entity. Another challenge is that the data is produced every second. Therefore, the amount of data is very huge and we need a mechanism to cache it and process efficiently. We design the distributed system, which allows the parallel processing and organizing of data on multiple machines. Additionally, it provides several extended function. First is the system level security mechanism, which allows the users from different enterprises to manage the identity verification easily. Second, it provides the workflow engine, which can build up the information process flow quickly according to the requirement. Third, it is integrated with a built-in web server to support the construction of B/S application framework. Fourth, it deploys several application modules according to the specific business, including web server, restful web service module with micro service architecture, shared storage module and oData protocol adaptation module.

Fig. 4 The architecture of Environment monitor application

7 Experiment

Before we adopt this application middleware platform to the system of environmental protection agency, we validate the platform with the true data samples from environmental protection agency. We take 12 days’ data sum up to about 1.2 million records, about 100000 records per day averagely. To compare the performance of middleware, we adopt these samples to RabbitMQ_AMQP, ActiveMQ_STOMP, RabbitMQ_STQMP and our proposed method ServiceBus_MidWare. We first monitor the cost of message
enqueueing and dequeueing when adopt the four different methods and the messages amount is $20000 \times 1024$ bytes. The result is shown in Fig.5. Y axis represents time spent.

![Fig.5 cost of enqueues & dequeues 2×10^4×1024 bytes](image)

We can see here, the performance of service bus middleware is better than ActiveMQ_STOMP and RabbitMQ_STOMP. Although there is a certain gap between the RabbitMQ_AMQP and our proposed method ServiceBus_MidWare, the integration of ServiceBus_MidWare with the IoT is more excellent. For the extended modules and the sound AOP-style construction, it can provide more run time information in detail. As the number of distributed nodes increases, ServiceBus_MidWare is about 3 times that of RabbitMQ. The result is shown in Fig.6.

![Fig.6 Throughput of ServiceBus_MidWare & RabbitMQ](image)

After the simulation test of sample data, we adopt this middleware to the system of environmental protection agency. As shown in Fig.7 and Fig.8, the monitor system based on the proposed middleware
platform is able to access heterogeneous data sources flexibly and develop the different displays of data. This system provides the government decision with important real time environment data base.

As we took 12 days’ data as samples, the Fig.9 shows the tcp requests with active responses of monitor data. We can see that the requests are always being blocked or pending and the system is busy to deal with accumulated requests after several hours, which causes the monitor delay. The Fig.10 is the system based on the proposed application middleware with service bus, we can clearly see that the number of tcp requests with active feedback is average in each period. The max number and the min number of requests succeeded in different period have no much difference and the color of the whole figuration is even, which suggest that there are less blocks in the run time of this improved system.

![Fig.7 The analysis page of toxic gases](image)

![Fig.8 The setting of Wastewater index](image)
8 Conclusion

According to the characters of real-time, dynamic scalability and uncertainty, we designed and implemented a multi-source application middleware based on service bus, the foundation of which is the three-layer model of Cyber-Physical system. This platform provides the external with a series of services encapsulated by all kinds of complex functions. Dynamic variable data of business systems are represented by the real-time redefined data entity, which satisfies the demands of comprehensive information process system of IoT time. According to the application of Shanghai Pudong environmental protection agency system, we validate the performance of the platform, which can provide solid support for the development of business application and has excellent safety and flexibility.

References

[1] EDWARD A. LEE. Cyber-Physical Systems - Are computing foundations adequate? [R]. Position Paper for NSF Workshop On Cyber-Physical Systems: Research Motivation, Techniques and Roadmap October 16-17, 2006, Austin, TX.

[2] G. TRIPATHI, D. SINGH. EOI: Entity of Interest Based Network Fusion for Future Internet Services[C]. ICHIT2011, September 23-25,2011, Daejeon, Korea.© Springer-Verlag Berlin Heidelberg, CCIS, vol. 206, pp. 39–45, 2011.

[3] EDWARD A Lee. Computing Foundations and Practice for Cyber-Physical Systems: A Preliminary Report.[R]. Technical report, University of California, Berkeley, UCB/EECS-2007-72, May, 2007.

[4] L. ATZORI, A. LERA, G. MORABITO. The Internet of Things: A Survey.[J] Computer networks. 54(16), pp. 2787–2805, 2010.
[5] KOTIS, K., Katasonov A. Semantic Interoperability on the Web of Things: The Smart Gateway Framework.[C] In Proceedings of the Sixth International Conference on Complex, Intelligent, and Software Intensive Systems Palermo, 2012.

[6] M. YUN, B. YUXIN. Research on the architecture and key technology of Internet of Things (IoT) applied on smart grid.[C] Advances in Energy Engineering (ICAEE). (2010) 69–72.

[7] J. SHI, X. YE, L. CHEN, P. ZHANG, and N. JIANG. ESF - An Extensive Service Foundation from Internet of Things Perspective[C] . 2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), pp. 59–64.

[8] Wei Xue-song, Zhang Yu-ping. The study and design of IOCframework.[M] 《The development of Computer technology》2006-3. 213-216.

[9] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, Spark: Cluster Computing with Working Sets., HotCloud 2010, 2010.