ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV032

DESIGN OF A COMPONENT-ORIENTED DISTRIBUTED DATA INTEGRATION MODEL

Z. G. Ni†, L Li, J. Luo, J. Liu, X. W. Zhou, Institute of Computer Application, China Academy of Engineering Physics, Mianyang City, China

Abstract

The control system of large scientific facilities is composed of several heterogeneous control systems. As time goes by, the facilities need to be continuously upgraded and the control system also needs to be upgraded. This is a challenge for the integration of complex and large-scale heterogeneous systems. This article describes the design of a data integration model based on component technology, software middleware (The Apache Thrift*) and real-time database. The realization of this model shields the relevant details of the software middleware, encapsulates the remote data acquisition as a local function operation, realizes the combination of data and complex calculations through scripts, and can be assembled into new components.

INTRODUCTION

Large scientific experimental devices generally consist of dozens of heterogeneous systems, each of which is also an independent control system. The control system architecture of a typical large scientific facilities is a two-tier architecture consisting of a monitoring layer of the network structure and a control layer of the fieldbus structure. The monitoring layer is deployed on the virtual server and the console computer to provide centralized operations for control, status, and data storage. The control layer is deployed on the virtual server or embedded controller to provide real-time collection and control of the device.

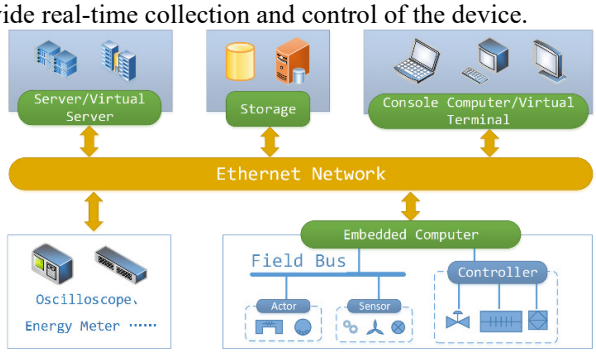

Figure 1: The control system architecture of the typical large scientific facilities [1].

As provided in Fig. 1, the monitoring layer is a software system based on Ethernet structure. It consists of a network switching system, a server system, and a console computer. It provides system services and human-machine interfaces for the facility control system, including control, monitoring, and data management.

At present, the integrated control system mainly used by large-scale laboratories and scientific research institutions around the world is EPICS or TANGO. In the automation industry, excellent instrument manufacturers will not only provide corresponding secondary development libraries, but also corresponding EPICS interfaces or TANGO Devices.

EPICS[2] is a set of software tools and applications which provide a software infrastructure for use in building distributed control systems to operate devices such as Particle Accelerators, Large Experiments and major Telescopes. Such distributed control systems typically comprise tens or even hundreds of computers, networked together to allow communication between them and to provide control and feedback of the various parts of the device from a central control room, or even remotely over the internet.

Tango[3] is an Open Source solution for SCADA and DCS. Open Source means you get all the source code under an Open Source free licence (LGPL and GPL). Supervisory Control and Data Acquisition (SCADA) systems are typically industrial type systems using standard hardware. Distributed Control Systems (DCS) are more flexible control systems used in more complex environments. Sardana is a good example of a Tango based Beamline SCADA.

Of course, many earlier control system software was built using CORBA software middleware, or implemented using the TCP custom protocol.

QUESTION

A few years later, the scientific experimental device will inevitably bring about the continuous upgrading of the system. How to adapt the control system and software to this change is a great challenge. The control system software constructed first needs to have a certain degree of scalability. Secondly, as technology evolves, the control system software needs to have a certain degree of compatibility; the early and current systems need to communicate and communicate, how to realize the communication between the early and current systems is a technical problem, which is a problem that this article needs to solve.

Another feature of the scientific experimental device is that the operator's demand is unstable. Users will constantly adjust their requirements based on their experience. We have to modify the code, compile, and test run, and the debugging time for software developers is very short, so the skills of software technicians are very high. If you can adapt to changes in requirements with zero programming, this is a great thing.

Therefore, we propose a component-oriented distributed data integration model, which is used to construct an implementation method of data acquisition and control from the device to the user interface.

In the following chapters, we first introduce the technologies that may be involved.

† email address: drops.ni@caep.cn.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV032

RELATED TECHNOLOGY

RPC and EDA: Thrift or Tango

Remote Procedure Call (RPC) is an interprocess communication technique. It is used for client-server applications. RPC mechanisms are used when a computer program causes a procedure or subroutine to execute in a different address space, which is coded as a normal procedure call without the programmer specifically coding the details for the remote interaction. This procedure call also manages low-level transport protocol, such as User Datagram Protocol, Transmission Control Protocol/Internet Protocol etc. It is used for carrying the message data between programs. The Full form of RPC is Remote Procedure Call.[4]

Event-driven architecture is a design model that connects distributed software systems and allows for efficient communication. EDA makes it possible to exchange information in real time or near real time. It is common in designing apps that rely on microservices. The concept of event-driven architecture is mainly realized through the publish/subscribe communication model. Publish/subscribe is a flexible messaging pattern that allows disparate system components to interact with one another asynchronously. The key point here is that pub/sub enables computers to communicate and react to data updates as they occur. This is in contrast to the traditional request/response messaging pattern where data is updated at intervals, as a response to a user-initiated request. There are always two participants — a client and a server. The client makes a call over the HTTP protocol and waits for a server to respond with the requested content.

Thrift[5] is a lightweight, language-independent software stack with an associated code generation mechanism for RPC. Thrift provides clean abstractions for data transport, data serialization, and application level processing. Thrift was originally developed by Facebook and now it is open sourced as an Apache project. Apache Thrift is a set of code-generation tools that allows developers to build RPC clients and servers by just defining the data types and service interfaces in a simple definition file. Given this file as an input, code is generated to build RPC clients and servers that communicate seamlessly across programming languages, as provided in Fig. 2.

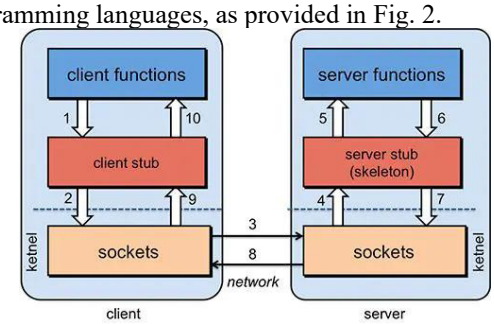

Figure 2: The software architecture of Thrift

Tango Controls is a free open source device-oriented controls toolkit for controlling any kind of hardware or software and building SCADA (supervisory control and data acquisition) systems. Tango Controls is operating system independent and supports C++, Java and Python for all the components. Tango Controls is a hardware independent toolkit. That means you can use your driver to connect hardware with Tango Controls, as provided in Fig. 3.

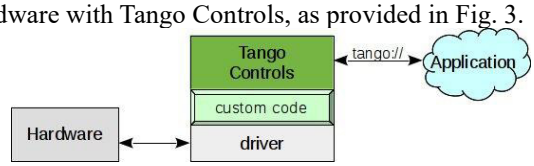

Figure 3: Tango Controls can be used as a distributed system

Script Engine: Elk[6]

Elk is a tiny embeddable JavaScript engine that implements a small but usable subset of ES6. It is designed for microcontroller development. Instead of writing firmware code in C/C++, Elk allows to develop in JavaScript. Another use case is providing customers with a secure, protected scripting environment for product customisation.

Realtime Database: LevelDB[7]

LevelDB is a fast key-value storage library written at Google that provides an ordered mapping from string keys to string values. LevelDB has three basic operations: Get, Put, and Delete. Get retrieves a value given a key, Put writes a value into a key, creating the key if it doesn't exist, and Delete deletes the key and its value. There are open (takes a filename argument) and close functions for creating/loading and unloading a database, and functions that return iterators over all the keys and values. The keys and values can be any byte array and not just strings. This is useful if you have data that you want to store that you don't want to encode into a string. LevelDB supports atomic operations. You can run many operations at once in a single uninterruptible call.

COMPONENT-ORIENTED DISTRIBUTED DATA INTEGRATION MODEL

Component of Integrated Control And Data Acquisition

As provided in Fig. 4, CICADA is the Component of Integrated Control And Data Acquisition. CICADA implements a common component for data acquisition and integrated control. Data is stored in a real-time database, control logic is stored in JavaScript scripts, and the script engine is used to update data and dispatch commands regularly. It can exist as a distributed service in the local area network and be called by other component instances; it can also be used as a service access module of the client software to access other component instances; it can also be used as a server and implemented as an IO device control server; Replacing different RPC modules can be used as a bridge between different middleware software.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV032

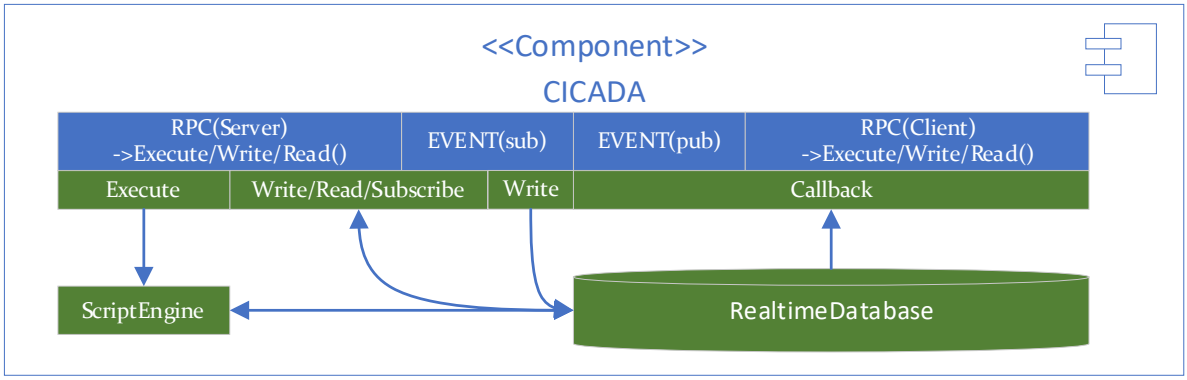

Figure 4: The composition structure of Component of Integrated Control And Data Acquisition.

The Data Centre (Realtime Database) is the data sharing centre of the components, which is responsible for storing and interacting data. The data tag can be configured with multiple attributes, what kind of behaviour is triggered after the data is executed or updated, whether the data is readable and writable, and so on.

The script engine is responsible for interpreting the executor to execute the script, interacting with the Data Centre, and realizing the logical operation of the data.

The local function set is responsible for basic functions such as executing scripts, reading and writing data, subscribing to data, and publishing data.

The RPC/EVENT module is responsible for information exchange between components, including RPC and event-driven methods, which can be used according to the application scenario, or both. The RPC/EVENT module is compatible with TANGO and Thrift software middleware, and supports expansion to CORBA or other software middleware products.

The Entity of CICADA

CICADA has three forms of existence, one is to be integrated by the client software in the form of a dynamic library, as the data acquisition and integrated control module of the client software; the other is to exist as a system service and be dispatched by the client software as a device service, and at the same time As an example of the client software scheduling other components; one way is to exist as a device service, which can be accessed by system services or client software, and part of the RPC module is replaced by the hardware driver SDK.

When it exists as a service, we design a container for CICADA. It contains an administrative module that is responsible for management functions such as creating, restarting and deleting components, similar to the DServer module in Tango software middleware.

As shown in the Fig. 5, we usually divide the control software into three layers [8]: integrated control layer, system service layer and equipment service layer. In these three-layer control software, we can all use CICADA to realize the interconnection and intercommunication part of the software at their respective levels, which greatly simplifies the many-to-many access control function between nodes without worrying about the implementation details.

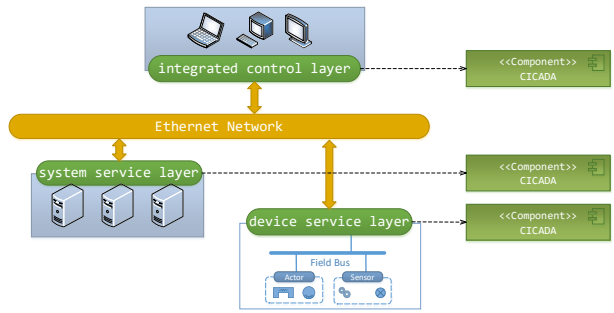

Figure 5: Use CICADA in the three-tier control software architecture.

Connection between CICADA Entities

CICADA entities exchange information and control scheduling through software middleware, as provided in Fig. 6. Components run in their own containers, which can be servers or clients. As an independent microservice or client, the CICADA entity acts as a node in a distributed network. The corresponding connection relationship is established according to the business scenario. They are loosely coupled, and the update influence domain of the component is limited, which can be guaranteed The control system software of the entire facility can evolve forward relatively stably and smoothly.

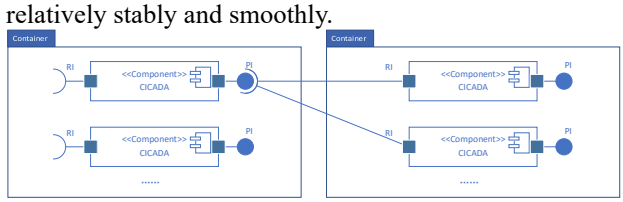

Figure 6: Exchange information and control scheduling between CICADA Entities.

Connection between early and current control systems

For example, the new system is implemented using TANGO software middleware, while the early system is implemented using Thrift software middleware. How to realize the information exchange between the early software system and the new system is a problem.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV032

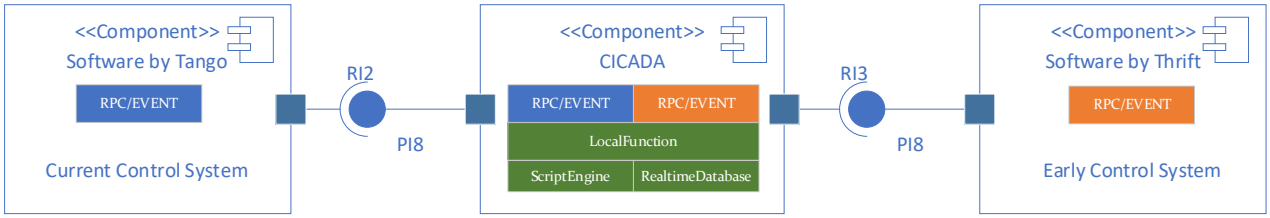

Figure 7: Use CICADA as a bridge between early and current control systems.

Now we use CICADA as a bridge between the two systems, basically zero programming can realize the information exchange function between the two software systems, as provided in Fig. 7.

CONCLUSION

Through the design and realization of CICADA components, we achieved two goals: the interconnection between the early software system and the current software system, and zero programming to adapt to the rapid changes in software requirements. Of course, we implemented this component to adapt to a workaround under specific needs. Whether it is effective for a long time still needs time to test, and this component is not considered in terms of QoS, so it needs to be strengthened in the follow-up work. Hope to communicate with colleagues, we will make CICADA components better and more practical.

REFERENCES

[1] D. J. Yao *et al.*, "Research on Software Architecture of Centralized Control System for High Power Laser Facilities, Computer Engineering and Design", vol. 28, pp. 1737-1740, 2007.

[2] EPICS Control System, http://www.aps.anl.gov/epics/

[3] TANGO Control System, http://www.tango-controls.org/

[4] RPC, https://www.guru99.com/remote-procedure-call-rpc.html

[5] Thrift, https://thrift-tutorial.readthedocs.io/en/latest/in-tro.html

[6] ELK, https://github.com/cesanta/elk

[7] LevelDB, https://github.com/google/leveldb

[8] Z. G. Ni, L Li, J. Luo, J. Liu, X. W. Zhou, "The Design of Intelligent Integrated Control Software Framework of Facili-ties for Scientific Experiments", in Proc. *17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, New York, NY, USA, Oct. 2019, pp. 132-136, doi:10.18429/JACoW-ICALEPCS2019-MOMPL007