Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2022.030304
ISSN 2790-0916 Vol. 3, Issue 3: 26-35

Distributed System Based on Software Components and Middleware Technology

Kothapa Lakshmi*

Loughborough University, Leicestershire, England

**corresponding author*

Keywords: Software Components, Middleware Technology, Distributed Systems, Distributed Testing

Abstract: With the wide application of Internet technology, more and more application systems use a dynamic and open network environment as a platform for computing and operation, and the functional requirements of the system are becoming more and more complex. Due to its strong real-time, openness, scalability and loose coupling characteristics, distributed component-based systems have been widely used in many fields such as civil industrial systems and ship equipment. The main purpose of this paper is to study the design and implementation of distributed systems (DS) based on software components (SC) and middleware technology (MT). In this paper, through the in-depth analysis of the MT, the middleware that conforms to the automatic testing technology is developed, and the instrument control, process communication, data management and other services of the middleware are realized, and the interaction between the application programs and the platform are tested. Experiments show that the average transmission efficiency of the new system using MT for data integration can be almost 8 times higher than that of the original system. After testing with data (the maximum amount of data per day does not exceed 20,000), when the supervision server is idle, the time required to transmit data every day does not exceed 4 minutes, which can basically guarantee the normal operation of the system.

1. Introduction

With the advancement of computer technology, business information systems have also made great progress than in the past. Enterprise-level applications are no longer satisfied with stand-alone systems and simple server-side systems, but are moving towards a multi-level shared environment. In fact, the idea of using distributed components to integrate various application systems in a distributed environment has existed in the computer industry for a long time, but it has not been realized. One of the main reasons is the lack of standards for MT [1].

In a related study, Griffin et al. proposed a novel distributed stack composition verification method to verify each component based only on the specifications of the lower components [2]. The robustness of the temporal logic of components and the reduced transformation of operational semantics with respect to a distributed stack of components is demonstrated. demonstrated an experimental method for selecting a set of SCs based on computational experiments that simulate the desired operating conditions of the software system under development [3]. A mathematical model is constructed designed to efficiently select components from the available alternatives. Albarrak introduced Trust But Verify (TBV) middleware [4], TBV intercepts messages between senders and receivers to verify the consistency of messages according to rules related to the message type. Depending on the verification, the message is either delivered to the recipient or blocked. Even if components authenticate each other, it is possible for their counterparts to malfunction or behave maliciously, convincing receivers to take harmful actions.

Due to the development of MT, the software architecture has undergone major changes. In the traditional two-tier C/S software structure, a middleware layer is added to make it a three-tier C/S software structure to build a DS. Therefore, this paper proposes a distributed testing (DT) technology based on MT to solve the problems of the current automatic testing system. Through in-depth analysis of MT, middleware that conforms to automatic testing technology is developed, and services such as instrument control, process communication, and data management of middleware are realized. The middleware provides an external operation interface suitable for advanced test equipment to determine the TPS motion; the middleware protects the compatibility between the underlying applications, realizes the interaction between the applications and the flexibility and extensibility of the test platform; the middleware Provides the ability to call remote process functions and perform remote management and error checking, enabling remote testing and troubleshooting.

2. Design Research

2.1. Application of MT in the System

After introducing the middle layer in a multi-data source system with traditional C/S structure [5-6], the data flow in the system will change. Figure 1 clearly shows this change:

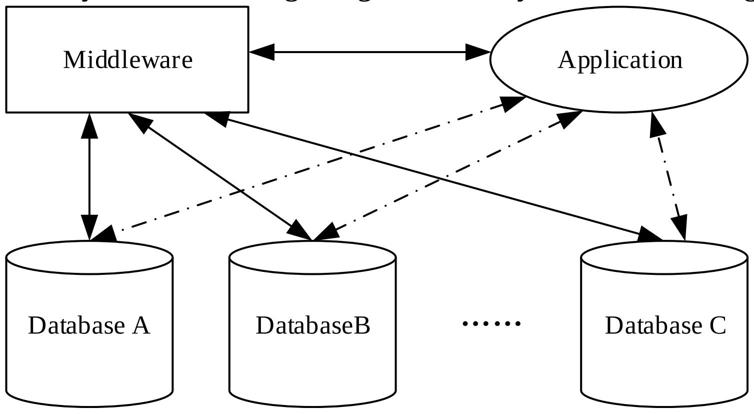

Figure 1. System structure with the introduction of MT

The dotted line in the figure shows the data flow when no middleware is used, and the solid line shows the data access flow after adding middleware [7-8]. After the middle layer introduces the data integration system, it brings many benefits to the system:

(l) Transparency: As can be seen from Figure 1, in the old data access system, the data link is complex; in the system with the introduction of the middle layer, no matter whether the application faces one data source or multiple The data source, regardless of the storage location of the database and the data mode, is the same for the application, and the application needs to be modified. When the data source migrates or the structure changes, only the middle layer needs to be modified. The corresponding components can be assembled.

(2) Security: In the old data access system, because the application program accesses the database directly, that is to say, the application program needs to master the access authority of the database, so the security of the system is subject to certain threats. After adopting the MT, the application system will not have the opportunity to communicate directly with the database because it only has a relationship with the middleware, and the data security is naturally improved [9-10].

(3) Transaction management and concurrency control: Transaction control is a very important issue in database-based information systems, and it must be ensured that the data submitted in the database is complete and correct. In addition to the control on the database side, in the data integration system, because of operations involving multiple data sources, a global transaction management and conflict handling mechanism is required. In the data integration system that adopts MT, the work that needs to coordinate and unified management of each data source is realized by middleware, which greatly reduces the complexity of the application [11-12].

2.2. Main Functions of DS Software

The main function of the distributed test system software based on middleware is to test and diagnose the device under test; it can manage all kinds of information of the device under test and save it for a long time, which is convenient for testers to manage data and reduce the management of testers. The difficulty of data; the ability to perform long-distance testing of the device under test and avoid extreme environmental testing such as high electromagnetic radiation [13-14]. The detailed analysis is as follows:

(1) Remote operation control: When the test system user cannot test or diagnose the problem of the equipment under test, the developer of the test system can operate and control the equipment of the user through the test middleware and network in the system for testing. Quickly and accurately give the maintenance strategy of the device under test.

(2) Data management: It can manage the information of the device under test, manage the test result data, fault diagnosis data maintenance information according to the category of the device under test, and realize the access to the test data anytime and anywhere. The software has the function of data management [15-16].

(3) Testing information sharing: During the testing process, resources such as the object under test, testing system, and support equipment are not in the same distributed network, and information sharing cannot be achieved, while DT systems can operate and communicate with each other, as far as possible. Enhance the utilization of test resources and realize test information sharing.

(4) Instrument control management: The distributed test system can control different types of instruments and equipment, operate the entire test process, complete the automatic test of the equipment under test, allocate instrument resources, and integrate distributed instruments from different manufacturers. The middleware of the test system must be able to control the instrument and provide a common instrument driver interface [17-18].

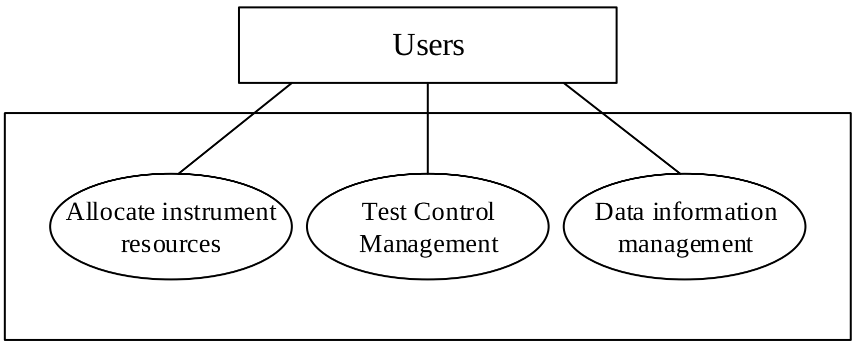

Figure 2. User use case diagram

2.3. System Index Analysis

The spatial resolution determined by the narrowband light pulse can be denoted as Rp, and its expression is:

$$R_p = \frac{t_w V_g}{2} \tag{1}$$

where tw and Vg are the pulse width and the group velocity of light, respectively. Since the group velocity is known, it can be calculated as:

$$R_p = \frac{t_w}{10} \tag{2}$$

And the spatial resolution RA determined by the bandwidth B in the photodetector is:

$$R_A \approx \frac{100}{B} \tag{3}$$

The expression of the spatial resolution / RA D determined by the A/D conversion speed f in the capture card is:

$$R_{A/D} \approx \frac{100}{f} \tag{4}$$

From the above formula, the spatial resolution R of the system can be obtained as:

$$R = \sqrt{R_P^2 + R_A^2 + R_{A/D}^2} \tag{5}$$

When determining the pulse width of the laser, on the one hand, the pulse width should be guaranteed to be small enough, and on the other hand, the problem of low pulse energy and reduced system signal-to-noise ratio should be considered. Therefore, it is necessary to choose the pulse width reasonably. The pulse width of the light source used in this system is 5 ns, which can improve the spatial resolution of the system. The frequency band width of the photodetector is 3dB and the bandwidth is 150 B MHz.

3. Experimental Study

3.1. Software Scheme Design

The distributed test platform based on middleware provides middleware interface functions for the upper-layer test program to call. For the upper-layer test program, the realization of remote testing, network communication, and data management are all transparent, that is to say, for it is a local test for a single processor. This design is based on testing hardware resources. The software design adopts RPC middleware communication technology, automatic testing instrument control technology and database technology to realize data display, data analysis, instrument control, resource allocation, information scheduling, and data storage of the test system. and other functions.

The middleware software provides various application services, such as: instrument control agent service, transaction processing service, data management agent service, logic application agent service, etc., and then handles various tasks transmitted by the test application layer in the form of an agent. The middleware software is based on the modular design idea, and each part of the software is divided into middleware modules with corresponding functions. The distributed test platform assembles each module in a visual way to quickly test and diagnose the equipment under test. The distributed test system is designed layer structure, as shown in Figure 3.

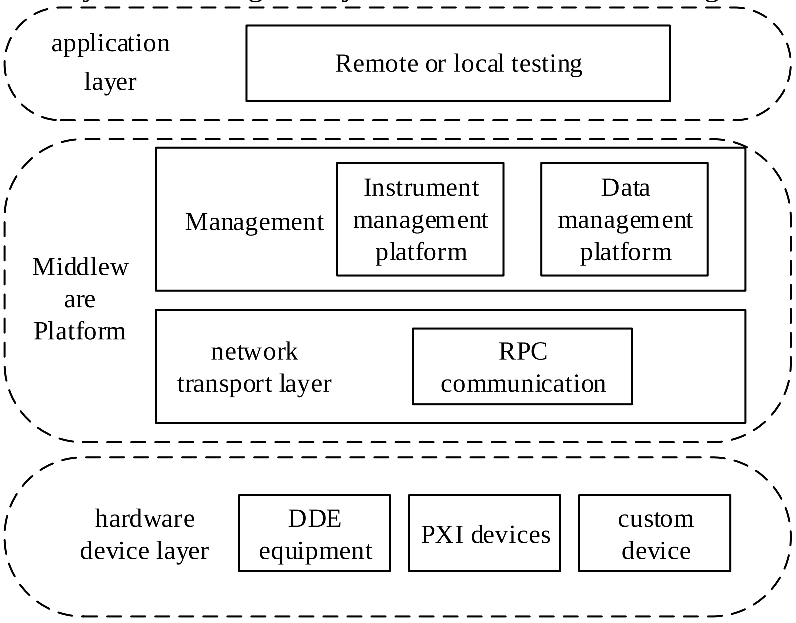

Figure 3. Software framework diagram of distributed test system based on middleware

As can be seen from the figure, the structure conforms to the three-tier C/S mode, the middleware is the middle layer, and plays a key role in the DT system. The middleware builds the DT basic framework model. For upper-layer applications, interfaces for various testing services are provided, so that the lower layer can be abstracted and encapsulated to cover as many software platforms as possible. The test software has the following features:

(1) Satisfy the timeliness of the response. For the test terminal, the faster the response, the better, instead of transmission delay and avoiding long waits.

(2) To meet the stability of the test program, for the test system, it may be necessary to continuously test and diagnose the faults of the equipment under test with high requirements, and give accurate test results.

In order to meet the above characteristics, the whole distributed test system is divided into three layers, but it is actually a four-layer structure, namely the hardware device layer, the network transmission layer, the management layer, and the application layer. This paper mainly studies the middleware platform including the network transport layer and the management layer, that is, the management operation platform.

(1) Network transport layer: It adopts RPC communication middleware, shields the details of

network communication, selects the working mode of non-blocking communication, realizes the communication program of client/server, and can receive both TCP connection requests and UDP according to actual needs. The connection request and provide the corresponding service request, this part will analyze the design in detail in Chapter 4.

(2) Management layer: It is divided into instrument management and control module and data information management module. This part is analyzed and designed in the third and fifth chapters in turn. The instrument management control module analyzes the actual instruments used in this project, realizes the modular driver package and encapsulates it into modules with good reusability and maintainability, and finally can efficiently carry out program control operations. The data information management module uploads and saves the data of each test and the relevant information of the device under test to the database.

3.2. Overall Design of System Software

Software requirements: The distributed optical fiber temperature measurement system is often used in the temperature monitoring of the outdoor environment. The software of this system needs to have an intuitive display of the monitoring area, showing the position of laying the temperature measurement cable and the corresponding temperature. Secondly, in order to meet the viewing and analysis of the temperature data of the monitoring area in the engineering project, the software needs to be able to store the temperature data. At the same time, in order to facilitate the use of the staff, it is necessary to expand around the main functions of the system and develop an interface that is convenient for users to operate.

The realization principle of the software: After the requirement analysis in the previous part, it is necessary to develop a software with a graphical interface. Therefore, the first step is to select a development platform. VISUAL C# is more convenient for graphical interface programming among many development platforms. It is very convenient to use WPF (Windows Presentation Foundation) interface programming technology to develop graphical interfaces, so it has become the software development platform. means of technical implementation. For the storage and query of temperature data, a database is used as the basis for realization. Compared with traditional text storage, the database can store a larger amount of data and support various conditional queries. In the signal display interface, by using the display control that comes with VISUAL C#, the control that comes with the system has more complete functions than the control that you write yourself.

Functional design of software: In order to meet the above requirements, it is necessary to design software with multiple functions, which can be well combined with actual engineering projects. After design, the main functions of this software are: map file import, monitoring map display, temperature alarm, temperature data storage, historical temperature data query, data report generation, signal and temperature curve display, etc.

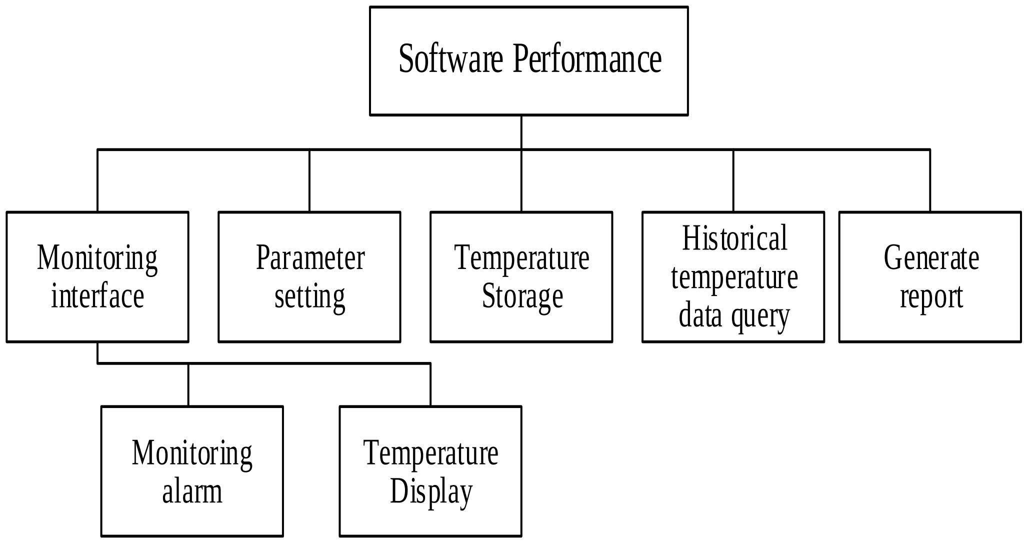

Figure 4. System block diagram of the software part

4. Experiment Analysis

4.1. Analysis of Integration Efficiency

When analyzing various indicators of the system, the operation efficiency is the first consideration. The original system takes at least 8 minutes to transmit a client's data, and the longest time takes 30 minutes, and it often fails. In the new scheme, the integration method based on MT is adopted, and measures to improve system efficiency such as data compression processing and incremental data extraction are added; however, some other measures that are not conducive to improving efficiency are also added for the performance of other aspects of the system. The following factors all have an impact on the efficiency of the system. Table 1 compares the factors that affect the efficiency in the system, and estimates the efficiency impact ratio:

Table 1. Efficiency comparison table between old and new systems

Impact factor	Original system	New system	New and old system efficiency ratio
Transfer data	Current month	Incremental	0.1
Data buffering mechanism	None	Have	0.75
Data compression	Have	Have	0.5
Data encryption	None	Have	1.5
Application server	None	Have	2
Final result			0.125

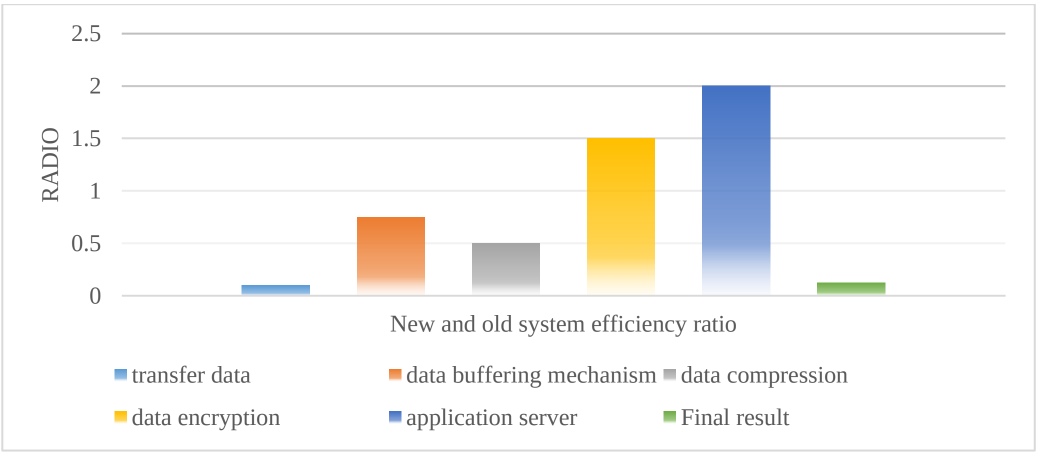

Figure 5. Comparison and analysis of the efficiency of the old and new systems

As can be seen from the data comparison shown in the figure above, the average transmission efficiency of the new system using MT for data integration can be almost 8 times higher than that of the original system. As an average estimate, it can be seen that the data transfer time is stable. After testing with data (the maximum amount of data per day does not exceed 20,000), when the supervision server is idle, the time required to transmit data every day does not exceed 4 minutes, which can basically guarantee the normal operation of the system.

4.2. Software System Testing

In order to verify the function of the test system, configure each middleware module, build a test platform based on middleware, write corresponding application layer programs, use black box testing, design corresponding test cases for testing, fault diagnosis and data information management, The system test results are shown in Table 2.

Table 2. Software system test results

Test case	Testing method	Expected outcome	Test results
Instrument and equipment interchange	Validation by replacing instruments from different manufacturers	Instrument interchange has no effect on upper-level test assemblies	Pass
Remote testing and troubleshooting	Click the application test button to run the test assembly offsite to test and troubleshoot the device under test	Basically similar to the local test results	Pass
Information sharing	Different test points, call each other equipment, equipment under test information and maintenance information, etc.	Different test information can be shared at different test points	Pass
Data information management	Click the import database button in the application testing interface	Store test results, maintenance information, etc. To the database, and display success	Pass

Through the above test methods, the main functions of the distributed test software based on middleware are verified. According to the test results, it can be concluded that the instrument control module, process communication module and data management module of the middleware can work together systematically to realize the instrument Management, local and remote testing, information sharing and management of data information, etc., show that the test software can work properly.

5. Conclusion

Middleware is the overall service between applications, operating systems and information systems. It has standard system interfaces and processes, and has specific implementations that conform to the standard interface and process specifications of different application platforms and operating systems. This paper mainly uses middleware to solve the distributed heterogeneous problem. In particular, it can meet the needs of a large number of applications, work on various application platforms and operating systems, support distributed computing, provide interoperability of applications or services across networks, devices, and device platforms, and support standard interfaces and programs. As an important layer between system software and applications, middleware abstracts away from typical shared application processes, leaving behind application-related business logic details and maintains key characteristics of typical distributed interaction processes. After abstraction, the complex DS is presented to the application in the form of a unified layer after refinement and special separation. Applications in the environment provided by middleware can better focus on business logic, and adopt the form of components to achieve good cooperation in different environments.

Funding

This article is not supported by any foundation.

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

Conflict of Interest

The author states that this article has no conflict of interest.

References

[1] Kaliukh Y I, Berchun Y A. Four-Mode Model of Dynamics of DS. Journal of Automation and Information Sciences, 2020, 52(2):1-12.
[2] Griffin J, Lesani M, Shadab N, et al. TLC: temporal logic of distributed components. Proceedings of the ACM on Programming Languages, 2020, 4(ICFP):1-30.
[3] Gusev A, Ilin D, Kolyasnikov P, et al. Effective Selection of SCs Based on Experimental Evaluations of Quality of Operation. Engineering Letters, 2020, 28(2):420-427.
[4] Albarrak R, Menasce D A. Trust But Verify: a framework for the trustworthiness of DS. IEEE Transactions on Dependable and Secure Computing, 2020, PP(99):1-1.
[5] Kudzh S A, Tsvetkov V Y, Rogov I E. Life cycle support SCs. Russian Technological Journal, 2020, 8(5):19-33.
[6] Spirovska K, Didona D, Zwaenepoel W. Optimistic Causal Consistency for Geo-Replicated Key-Value Stores. IEEE Transactions on Parallel and DS, 2021, 32(3):527-542.

[7] Prayogo Y, Bayu M. Validation of Technology Components for Peanut Validation Of Technology Components For Peanut Pod Borer (Etiella Zinckenella Triet.) Control. Jurnal Hama Dan Penyakit Tumbuhan Tropika, 2020, 20(1):1-12.

[8] Gnanasankaran N, Iyakutti K, Alagarsamy K, et al. The impact of COTS components on software quality in IT Industry: A Survey and its analysis. International Journal on Computer Science and Engineering, 2021, 4(1):1-10.

[9] Sakir R, Bhardwaj S, Kim D S. Enhanced faulty node detection with interval weighting factor for DS. Journal of Communications and Networks, 2021, 23(1):34-42.

[10] Saraswat B K, Suryavanshi R, Yadav D. Formal Specification & Verification of Checkpoint Algorithm for DS using Event - B. International Journal of Engineering Trends and Technology, 2021, 69(4):1-9.

[11] Imre S, Berces M. Entanglement-Based Competition Resolution in DS. IEEE Access, 2021, PP(99):1-1.

[12] Santos A A, Silva A, Magalhes A P, et al. Determinism of Replicated DS–A Timing Analysis of the Data Passing Process. Advances in Science Technology and Engineering Systems Journal, 2020, 5(6):531-537.

[13] Arun B, Peluso S, Palmieri R, et al. Taming the Contention in Consensus-based DS. IEEE Transactions on Dependable and Secure Computing, 2020, PP(99):1-1.

[14] Albarrak R, Menasce D A. Trust But Verify: a framework for the trustworthiness of DS. IEEE Transactions on Dependable and Secure Computing, 2020, PP(99):1-1.

[15] Tsigkanos C, Garriga M, Baresi L, et al. Cloud Deployment Tradeoffs for the Analysis of Spatially-DS of Internet-of-Things. ACM Transactions on Internet Technology (TOIT), 2020, 20(2):1-23.

[16] Silva E O, Vanco W E, Guimaraes G C. Capacitor Bank Sizing for Squirrel Cage Induction Generators Operating in DS. IEEE Access, 2020, PP(99):1-1.

[17] Abdukarimovich G N, Khudainazarovna K M, Ravshabekovna S S. Building models of territorial DS. International Journal on Integrated Education, 2020, 3(10):300-303.

[18] Secara I A. Challenges and Considerations in Developing and Architecting Large-Scale DS. International Journal of Internet and DS, 2020, 04(1):1-13.