Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2021.020107
ISSN 2790-0916 Vol. 2, Issue 1: 83-91

Design and Implementation of Distributed Resource Management and Platform Deployment Based on Logical Clock

Saravannan Sriddevi*

Tamale Technical University, Ghana

**corresponding author*

Keywords: Logical Clock, Distributed Systems, Resource Management, Platform Deployment

Abstract: With the rise of cloud computing technology and the continuous improvement and progress, at the same time, in the environment of technological innovation, the demand for information construction is becoming more and more urgent. The purpose of this paper is the design and implementation of distributed resource management and platform deployment based on logical clocks. After comparing container technology and virtual machine technology, this paper uses cloud computing-related technologies such as containers and orchestration tools to analyze scheduling strategies within the laboratory, platform design and service deployment to build a resource management platform. After studying the scheduling strategy, this paper designs, implements and deploys the platform and services. The hardware, network and other environments of the laboratory are configured and planned, and then the overall architecture of the resource management platform is designed. Latency test results for business deployments show that when the data grows significantly, the time taken by the system also increases, especially when the data goes from 500 MB to 1 GB. The delay test results of service unloading show that the size of the data does not have a great impact on the delay of unloading services, but the overall efficiency of the distributed resource management.

1. Introduction

With the improvement and development of information technology and digital manufacturing, the main purpose of information technology and digital manufacturing is to manage digital resources and ensure information services as auxiliary means. Management and improvement of the construction cycle is long. Only through long-term accumulation and precipitation, all participants can continuously supplement and improve the comprehensive digital resource library with rich

content and wide-ranging orientation in the digital resource management platform [1]. Therefore, in IT and digital construction, it is necessary to create an open, shared, clear and easy-to-use digital resource management with high security; at the same time, it can also improve recruitment and utilization [2].

Scholars at home and abroad have conducted various researches on distributed resource management and deployment platforms. Schwenkreis F proposed a method to convert collected team handball game data into a format that allows the use of taxonomy and methods to search for association rules. To be able to search for patterns at any time of a match, the concept of a logical clock is introduced, which becomes an important part of data preparation. The applied data mining method is described in detail using the RapidMiner process and its settings. However, this method is independent of the data mining tools used. Based on the results of the data mining process, the applicability of data mining techniques in a given context will be discussed. In particular, it will be shown that rule-based results have significant advantages over methods using SVMs in a given context. The results are also compared against logical clocks, which will show how the pattern evolves over time in the case of team handball [3]. Pozzetti T proposed several implementations of smart clocks, most recently Coded Vector Clock (EVC), a scalar Coded Vector Clock (VC) framework that uses prime numbers to improve performance and size. The concept of Resettable Vector Code Clock (REVC), a new EVC-based logic clock implementation, was proposed and developed in response to its extremely high growth rate. Show how to use REVC in shared memory and message passing systems to achieve stable computing clocks. Let's take the growth rate advantage of REVC over EVC as an example. Finally, the practical application of REVC to the problem of dynamic contention detection in a multithreaded environment is introduced. This tool compares with current VC-based hardware DJIT^+ to show how REVC can help achieve high vector clock performance [4]. Drozd introduces dynamic clock signal routing designed to avoid synchronization failures in computing systems such as FPGAs. The proposed design reduces power consumption and improves power steering control in FPGA systems. These benefits are due to improvements in the assessment and measurement of the corresponding power base. Power parameter analysis has proven to be a good solution for detecting synchronization faults that destroy smart circuits. The main advantage is that traditional in-circuit testing based on intelligent control can be used to detect hidden synchronization errors in safety-related systems. Two main types of energy-oriented monitoring are considered: energy consumption and concurrent energy source regulation, using temperature and current energy sensors, respectively. Experiments are performed on a real FPGA system with controlled synchronous optimization and computer-aided design (CAD) tools are used to calculate the reduced values of power parameters. The results show that the control of the FPGA system is limited when the clock signal is used for temperature and power monitoring [5]. To sum up, some research achievements have been made in the design and implementation of distributed resource management and deployment platforms at home and abroad, which is very inspiring and meaningful for the development of this research work.

The purpose of this paper is to obtain a distributed system resource management that can automatically deploy services in a cluster and automatically manage services. Based on the OCF framework, the platform develops and deploys services in the form of components based on existing OCF components. Based on the resource management of the underlying system, the platform adopts an extensible node allocation strategy for node allocation, deploys and uninstalls services in the form of components, and abandons the traditional way of deploying and deploying isolation. The services are tightly coupled, thereby reducing the cost and difficulty of operation and maintenance, improving the stability of cluster services, and improving the resource utilization

efficiency of the entire cluster. On these two subsystems, a distributed development platform is built. Automated deployment and management capabilities.

2. Research on Distributed Resource Management and Platform Deployment Based on Logical Clock

2.1. Clock Constraint Language - CCSL

The core concept of Clock Con-straint Specifcation Language (CCSL) is the logical clock. One instant corresponds to a tick of the clock. The logical clock c can be expressed as a two-tuple, where Ic represents the set of moments, which is a total order on Ic, and the subscript can be omitted when there is no ambiguity. Based on logical time, CCSL only cares about the order relationship between moments, not the exact physical time interval between two moments [6-7]. Obviously, workers are discrete, according to <. The specified order can be a natural number I. The time number in, c[k] represents the kth time. CCSL also provides three other operators for specifying moment relationships, <, = and #. < is a reflexive, transitive binary relation that can be used to express causality [8-9].

CCation provides a variety of clock constraints to illustrate the relationship between clocks. Clock constraints can be divided into clock relations and clock expressions. The former describes the relationship between two clocks. The description of the clock relationship by the clock constraint is based on the description of the time relationship of the corresponding clock. The clock constraints provided by CCTL can express a variety of computing communication rules, such as: there are both rules for smart computing and communication, and rules for asynchronous computing and communication [10-11].

2.2. Distributed File System

Distributed file system is one of the important technologies supporting storage computing environment. Distributed file system eliminates some failure points and performance bottlenecks by cooperating with multiple nodes, and aims to meet key characteristics such as high availability, high performance and high load [12-13].

Through the research on the shared cloud file storage system, the aim is to fully integrate the software and hardware resources of various application servers and computers embedded in the digital construction of schools, to achieve high-quality sharing of resources of each application server and computer, and to understand the distribution in storage. The HDFS definition of the file system has been modified to better lay the foundation for the management of internal cloud storage resources in colleges and universities, and provide a reliable and comprehensive architecture design and management platform. Cloud storage architecture. Since the construction of a digital resource platform must consider a relatively long construction time, the digital resource management platform decided to adopt HDFS as the basic system architecture [14-15].

2.3. Object Storage Swift

The Swift storage system has two basic modules, namely proxy server and storage server [16-17]. The proxy server is the interface between the storage system and the outside, which is responsible for forwarding user requests to the appropriate storage server, and the storage server is responsible for storing the data in the cluster. The storage server has three types of accounts, containers and

objects. Users have their own Swift accounts in the cluster, and multiple containers can be created under the account, and the containers complete the storage of objects. Swift completes the mapping of storage objects to their physical storage locations through Ring. The structure uses a consistent hashing algorithm and introduces virtual nodes to ensure that even when the number of nodes in the cluster changes, data movement can be reduced as much as possible. It can also ensure the balanced storage of data in the cluster. In Swift, accounts, containers and objects have corresponding rings [18].

3. Design and Research of Distributed Resource Management and Platform Deployment Based on Logical Clock

3.1. Platform Deployment Environment

The network environment of the laboratory consists of an intranet and an extranet. In the internal network, each teaching and research room uses the switch H3CS1224P-X to connect the personal computer. In order to isolate the network of different teaching and research offices, and uses the routing function of the three-layer switch UnicacaUN-21608K. In addition, each teaching and research room has a router connected to the external network. By configuring multiple IP addresses on the personal computer and configuring the back route on the server, users can access the internal network and the external network at the same time.

Table 1. Platform server configuration

	CPU	RAM	Storage	NIC	GPU
Master	2*INTEL E7-4870	256G	6T	4	0
Node 1	4*INTEL E7-4870	512G	6T	4	2*GTX2080
Node 2	4*INTEL E7-4870	512G	6T	4	2*GTX2080
Storage Node 1	2*INTEL E7-4870	256G	20T	4	0
Storage Node 1	4*INTEL E7-4870	256G	20T	4	0

The platform currently includes 5 servers, and the related configurations are shown in Table 1. One of them is the master node of the platform, two are equipped with GeForce GTX2080 graphics card, the server with strong performance is used as the working node Node, which is responsible for running the Pod, and the remaining two have sufficient storage to provide the storage and backup services of the cloud platform. The operating system Ubuntu is pre-installed on each server, and Dockerv and Kubernetesv1.20.1 environments are built. Each deployment is a heavy workload, so this article writes a shell script to automate deployment.

3.2. Platform Architecture Design

The resource management platform of this paper is divided into resource layer, management layer, application layer and presentation layer from top to bottom.

Figure 1. The resource layer is infrastructure resources, including physical host servers, storage and network resources; the management layer integrates some tools for monitoring and managing the system, such as the container orchestration framework Kubernetes; the application layer describes the services provided in the resource management platform, such as code warehouse Gitlab, warehouse manager Nexus, identity authentication LDAP, to complete the management of resources such as code, mirroring and personnel, as well as some heavy computing tasks (such as

deep learning) related services; the presentation layer is the user operation and use management platform The perform operations such as remote debugging, service management and visual resource monitoring. Services run on the platform as containers and are managed by Kubernetes.

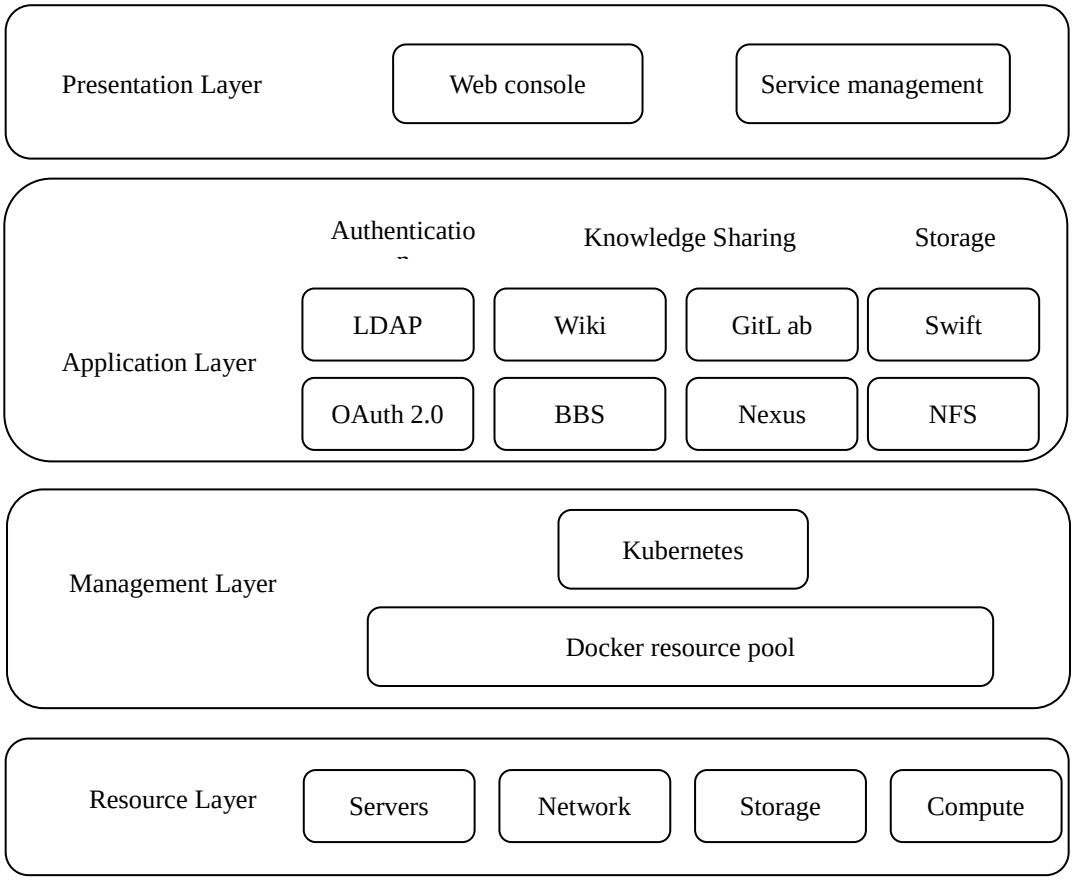

Figure 1. Resource management platform architecture

3.3. Clock Model

In electronic applications, clock components are often required to provide a periodic clock signal as their normal operating time. These clock elements typically use the piezoelectric effect of a crystal oscillator or the resonant frequency selection of an oscillator circuit to determine the clock frequency, ie clock speed. The number of oscillations is recorded by a counter and converted to the current clock value.

According to the characteristics of the above clock components, in the general case, the clock in the device can be modeled as:

$$\tau(t) = \frac{1}{f_0}\int_0^t f(s)ds + \tau(0) \quad (1)$$

where τ(t) is the hardware time of the node at the reference time t, fo is the operating frequency marked on the oscillator, f(s) is the actual operating frequency of the oscillator, and τ(0) is the clock value of the node at the initial moment.

When the device works in a stable external environment, that is, when the oscillation frequency

of the oscillator of the device clock part cannot be observed, the clock model of the node can be converted into a linear model according to equation (2) to change the relationship.

$$\tau(t) = \alpha t + \beta \tag{2}$$

where a is the clock skew that determines the rate at which the clock is running, and β is the clock rate that indicates the hardware clock skew.

4. Test Results and Analysis of Logical Clock-Based Distributed Resource Management and Platform Deployment

4.1. Delayed Testing of Service Deployment

Data file sizes are 0, 50 MB, 100 MB, 500 MB, and 1 GB. Four datasets are included with different data file sizes. During testing, the deployment performance delay between the system and Ansible is shown in Table 2.

Table 2. Latency test results for service deployment

data size	0	50MB	100MB	500MB	1GB
Ansible(s)	6.1	7.8	15.3	42.31	90.25
	6.4	7.6	15.2	43.21	92.36
	6.2	6.9	15.4	45.03	93.45
	5.9	7.2	14.9	43.36	89.67
Deploy Platform(s)	0.19	0.64	5.26	23.62	54.35
	0.17	0.68	5.43	24.35	53.45
	0.16	0.62	5.37	21.36	55.21
	0.16	0.69	5.18	23.45	54.36

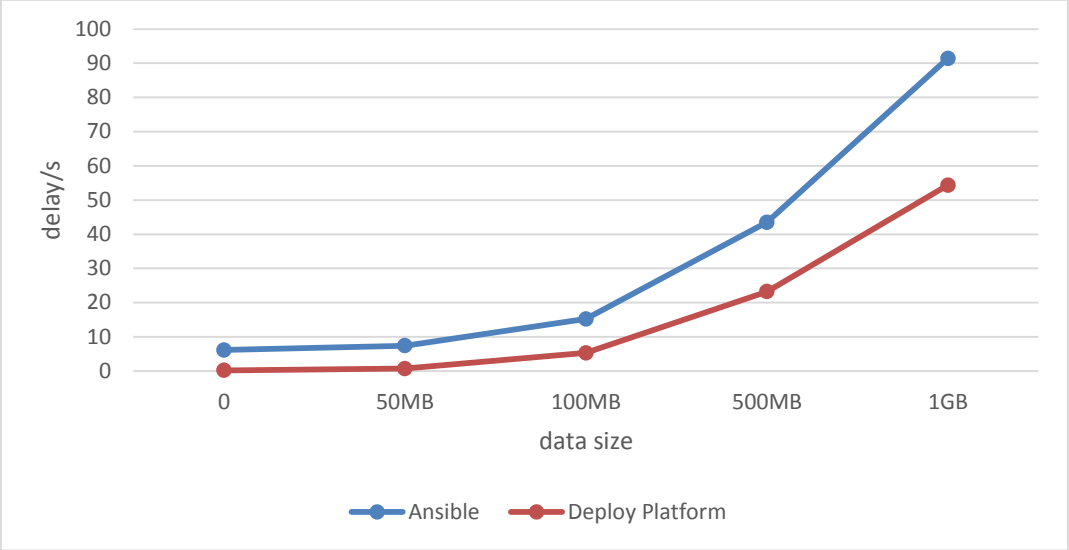

Figure 2. Service deployment average latency test results as data size changes

It can be seen from Figure 2 that when the data increases significantly, the time used by the program also increases, especially when the data changes from 500MB to 1GB.

4.2. Delay Test of Service Unloading

After each deployment is completed, we will also perform the corresponding uninstallation tasks, so that the subsequent deployment tasks can run smoothly, and count the delay of each uninstallation task. Here, 4 sets of data are selected, as shown in Table 3.

Table 3. Latency test results for service offload

Data size	0	50M	100M	500M	1G
Ansible(s)	4.2	3.7	3.6	4.2	4.6
	3.2	3.5	3.8	4.1	4.5
	3.3	2.9	3.4	3.8	4.8
	3.2	3.8	3.9	3.9	4.7
Deploy Platform(s)	0.052	0.053	0.062	0.089	0.112
	0.044	0.047	0.069	0.094	0.154
	0.046	0.062	0.068	0.087	0.167
	0.045	0.055	0.067	0.091	0.128

As can be seen from Figure 3, the size of the data does not have much impact on the latency of offloading services, but the overall efficiency of the resource management and distributed system development platform is higher than that of Ansible. In general, resource management and distributed development platforms perform well in terms of latency performance, so distributed system resource management and deployment platforms have good performance while fulfilling their own functional requirements.

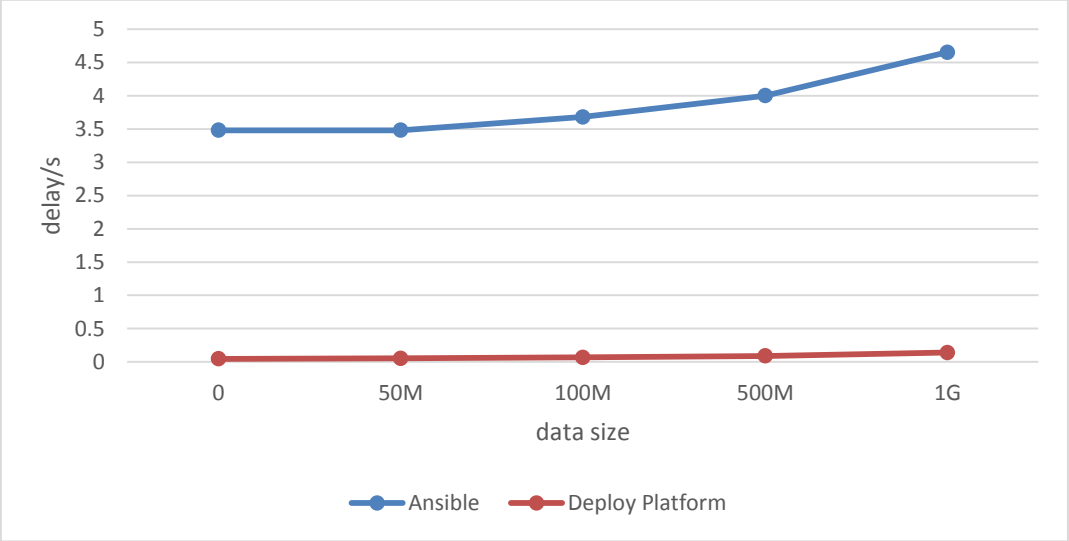

Figure 3. Service offload average latency test results as data size changes

5.Conclusion

A distributed process tree method is proposed to manage the network connected between devices, and on this basis, the real-time synchronization of source data and the interaction of data are considered. At the same time, synchronous resource management combines subscription/push resource data to synchronize only the required data changes in real time, and quickly respond to resource data changes under limited communication constraints. Aiming at the general resource management platform of the Internet of Things and the intelligent transmission scenario proposed, the solution and implementation of the general resource management platform to realize the platform architecture of the Internet of Things are given. Through performance comparison, it shows that the platform is efficient and timely in data management and programming.

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] Hoff J R. Conflux--An Asynchronous Two-to-One Multiplexor for Time-Division Multiplexing and Clockless, Tokenless Readout. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019, PP(99):1-13.

[2] Bagchi S. Design and topological analysis of probabilistic distributed mutual exclusion algorithm with unbiased refined ordering. Future Generation Computer Systems, 2019, 95(JUN.):175-186.

[3] Schwenkreis F. A Logical Clock Based Discovery of Patterns. International Journal of Data Science and Analysis, 2021, 7(4):98-108.

[4] Pozzetti T, Kshemkalyani A D. Resettable Encoded Vector Clock for Causality Analysis With an Application to Dynamic Race Detection. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(4):772-785.

[5] Drozd O, Nowakowski G, Sachenko A, et al. Power-Oriented Monitoring of Clock Signals in FPGA Systems for Critical Application. Sensors, 2021, 21(792):1-17.

[6] Kirichenko A F, Vernik I V, Kamkar M Y, et al. ERSFQ 8-bit Parallel Arithmetic Logic Unit. IEEE Transactions on Applied Superconductivity, 2019, PP(99):1-1.

[7] Gamayani U, Calista C, Ong A, et al. Cognitive Examination In Thalassemia Patients. The Open Psychology Journal, 2020, 13(1):95-100.

[8] Hameed M, Shakor H. Design and Implementation of Low Power Clock GatingTechnique in 16 bit ALU Circuit. Journal of Engineering and Applied Sciences, 2018, 13(9):2767-2772.

[9] Danielsson P, Postema T, Munir H. Heroku-Based Innovative Platform for Web-Based Deployment in Product Development at Axis. IEEE Access, 2021, PP(99):1-1.

[10] Wirasinghe C, Perera S, Ganepola D, et al. Frugal development and deployment of an innovative mobile health platform for COVID-19 in Sri Lanka: the case of SelfShield app. BMJ Innovations, 2021, 7(4):604-608.

[11] Iyer S. Automata processing in reconfigurable architectures: in-the-cloud deployment, cross-platform evaluation, and fast symbol-only reconfiguration. Computing reviews, 2020, 61(2):69-69.

[12] Galanis I, Anagnostopoulos I, Nguyen C, et al. Efficient Deployment of Spiking Neural Networks on SpiNNaker Neuromorphic Platform. Circuits and Systems II: Express Briefs, IEEE Transactions on, 2020, PP(99):1-1.

[13] Kim B. Big Data Platform Case Analysis and Deployment Strategies to Revitalize the Data Economy. Jouranl of Information and Security, 2021, 21(1):73-78.

[14] Santiago G V, Alvares A J. Deployment framework for the Internet of water meters using computer vision on ARM platform. Journal of Ambient Intelligence and Smart Environments, 2020, 12(1):35-60.

[15] Pedersen A, Valla M, Bofin A M, et al. FastPathology: An open-source platform for deep learning-based research and decision support in digital pathology. IEEE Access, 2021, PP(99):1-1.

[16] Aloi G, Fortino G, Gravina R, et al. Simulation-driven platform for Edge-based AAL systems. IEEE Journal on Selected Areas in Communications, 2020, PP(99):1-1.

[17] Hernandez M P, Mcfarlane D, Parlikad A K, et al. Relaxing Platform Dependencies in Agent-Based Control Systems. IEEE Access, 2021, PP(99):1-1.

[18] Sousa B M, Fonseca V, Cordeiro L, et al. EMPATIA: A Multichannel Platform for Participatory Budgeting. International Journal of Electronic Government Research, 2020, 15(2):58-89.