Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2020.010403
ISSN 2790-0916 Vol. 1, Issue 4: 18-24

Distributed System Computation Model Based on Mobile Agent Development Platform

Hassan Mumad*

Philippine Christian University Center for International Education, Philippines

**corresponding author*

Keywords: Mobile Agent Technology, Distributed System, Computing Model, Transparent Memory Algorithm

Abstract: With the advancement of Internet technology, people's requirements for data computing and processing are getting higher and higher, and it is difficult for a single computer to meet efficient task computing and data transmission. Therefore, multiple computers are connected to complete a large-scale project. The needs of the task are becoming more and more urgent. With the increase of this demand, the distributed system(DS) emerges as the times require, and the mobile agent technology provides a new way to solve the distributed computing(DC) problem. For the system proposed in this paper, the service quality of computing tasks is evaluated by the transparent memory(TM) usage effect of the DC system. The experiments show that the TM algorithm is the most robust to the amount of memory contributed by nodes; and then through the multi-queue node scheduling experiment, it is found that the low priority The response time of the task is higher than that of the high priority task.

1. Introduction

DC belongs to computer science. It divides a task covering a large amount of computing into multiple parts, transforms one task into multiple tasks, and assigns the divided tasks to multiple computers for processing. The result ensemble is the final result [1]. For the problem of a large amount of data, this task scheduling solution brings great convenience to people.

At present, the research on the DS computing model(CM) based on the mobile agent development platform has achieved remarkable results. Agents are widely used to implement typical stream computing tasks, users can upload work packages to the Agent platform and execute platform tasks. Since the data of the computing thread is continuously input, the computing process will continue until the user does not input data [2]. In scientific research and other fields, large-scale DC systems are used more and more widely and deeply. These systems make full use of the Internet network to build a DC system with fully heterogeneous system components in the WAN

Copyright: © 2020 by the authors. This is an Open Access article distributed under the Creative Commons Attribution License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (https://creativecommons.org/licenses/by/4.0/).

environment [3]. The DC model developed by a scholar on the Agent platform consists of many components, which interact with each other through the network. The system includes a three-layer structure of presentation logic, business logic, and database logic. Different structures have different functions and different processing tasks. Users input computing tasks into CMs, and output computing results under the interaction of various functional structures [4]. Although the DC system is widely used, in order to realize the task of CM or the function of dividing and scheduling, it is necessary to rely on the Agent to provide reliable technical support for it.

This paper first briefly introduces the basic concepts of the mobile agent development platform; then analyzes the theoretical model of the agent; then designs the architecture and functional modules of the DS CM based on the mobile agent platform; finally, through the TM experiment and multi-queue node scheduling experiments evaluate the system's quality of service for computing tasks.

2. Relevant Platform Theories and Models

2.1. Mobile Agent Development Platform

Agents usually run in an environment and have one or more targets. Mobile Agent platform is a brand-new DC model, and its superiority in computing massive data has been paid more and more attention [5]. Java language is very suitable for Agent programming, so many researchers choose Java language when designing Agent platform [6]. After years of development, the research on Agent theory and technology has made great progress and has become an important concept in current computer science.

2.2. Agent Theoretical Model

The agent is rational, which means that the agent can judge the effectiveness of the action according to its cognition of the world and its own knowledge, and execute the action it thinks is the most effective. For this reason, the agent usually needs to use a certain strategy to consider each candidate action [7]. In order to realize the rational subject, people have proposed a variety of agent architectures, among which the most eye-catching and widely used agent architecture is the BDI architecture. Some studies have transformed the BDI architecture into a formal theory and a software agent execution model based on beliefs, goals, and plans, and this model is used by the Jadex BDI inference engine [8-9].

The interaction between the agent and the environment can make the computing process adapt to the distributed environment, and the agent is formally abstracted. It is assumed that the environment is a set of finite discrete states, represented by a finite set E, where e1 and e2 are both subsets of the set E:

$$E = \{e_1, e_2, \ldots\} \tag{1}$$

Assuming that the agent can change the environment state through the executable task list, then the changed environment is represented by F, where f1 is the first action selected by the agent to act on the changed environment state, and f1 is the second action selected by the agent.

3. System Design

3.1. Architecture of DS CM

Figure 1 is the overall structure of the DS CM. In order to ensure the scalability of the system, the system adopts the method of dividing different logical computing domains. Participating nodes in a domain perform as many task sets as possible, and implement various load balancing and other algorithms in the domain [10]. A token is set in the domain to identify a domain server. Considering that the participating nodes in the domain may fail at any time or actively withdraw or the network is interrupted, the nodes in the same domain use most of the communication methods to confirm whether the domain server is healthy, and migrate the domain server to the domain server according to the situation. on other nodes [11].

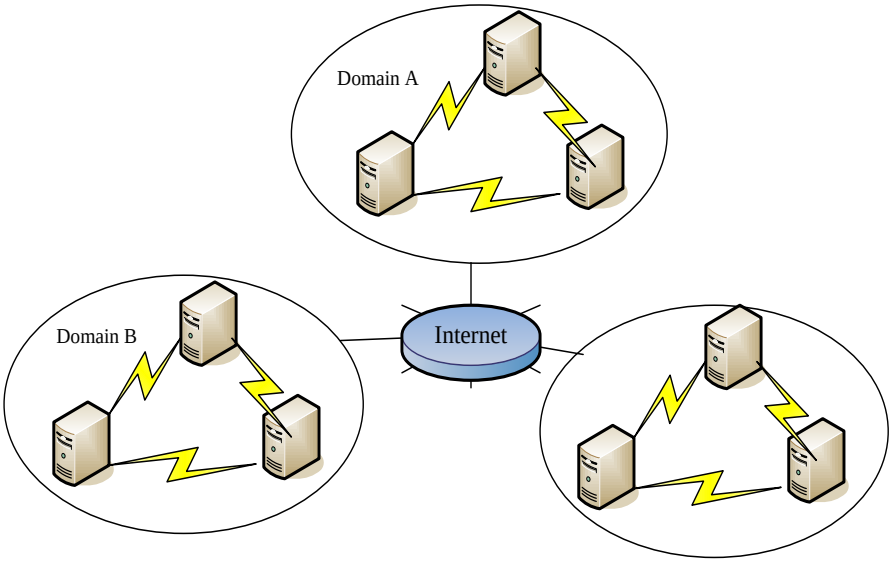

Figure 1. Overall structure of the DS CM

The DS CM is generally divided into four parts: the background database, the system server, the work machine that proposes work tasks, and the client computer that contributes to the calculation. The background database uses the ORACLE database to store the calculation result data returned by the client. These data are not simply stored. The system server will collect these data and analyze the correlation between them, and then generate new data based on the written data. The work branch is resubmitted to the user for operation [12-13]. Of course, the primary source of tasks remains researchers. The researchers submit tasks to the system server through the working machine. The server preliminarily analyzes the tasks and divides different sub-tasks to assign them to the client computing [14].

3.2. Functional Structure of DS CM Based on Agent Platform

Figure 2 shows the three modules of the DS CM, and their functions are described as follows:

(1) Session connection module

Mobile Agent development platform includes central nodes, event handlers, connectors and other important components. In addition to the event handler, a filter element can be added [15]. For clients and processing nodes, the most important thing is to establish a connection to the central

server, as well as install a connector and an event handler [16]. Each time a message signal is sent, a new session frame must be established to send the signal completely. After the message signal sending operation is completed, the current session frame needs to be closed, and then another session can be re-established to send other messages, that is, a session can only be satisfied once information sending [17].

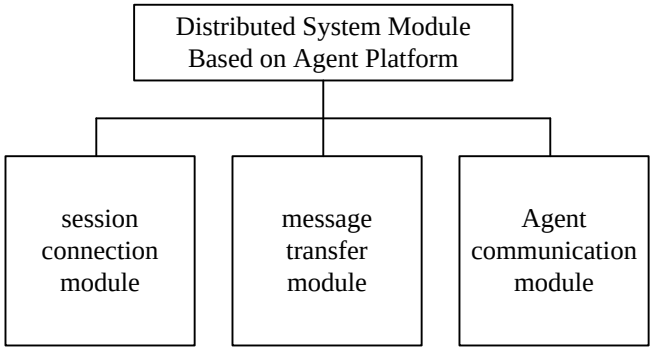

Figure 2. System function block diagram

(2) Message transmission module

The message module has a relay point for relaying messages between brokers. This relay point is functionally similar to a message queue for a DS, performing three functions: On the one hand, it sends messages or other data between brokers. By using this service, even if an agent moves in the action network, the agent's communication will not be interrupted; on the other hand, the relay point allows the agent to record messages or other types of data, and when the message or data arrives, the relay point Agents interested in such messages or data will be notified [18].

(3) Agent communication module

Agent exists in the system as an instance, which shows autonomy and sociality. We can consider coordinating the work of multiple agents as a group, and then we need to establish a communication language and communication medium [19]. If only one agent is running, and it does not run alternately with other agents, it will reduce the efficiency of the entire task processing mechanism, and its work efficiency will be very low. The key to solving this problem is to establish an agent communication mechanism, so that an agent can establish a connection with other agents through the communication mechanism when running alone, and form an interactive computing mechanism.

4. Evaluation of DS CMs

4.1. TM Experimental Evaluation

The goal of TM usage is to balance the division of memory among different types of applications, so as to ensure the transparency of local user applications when using the system to contribute memory resources. It allocates enough memory to local applications to achieve a transparent effect, and try to meet the memory requirements of high-priority tasks. The key question is how many memory pages are allocated to distributed applications to ensure that the operation of local applications is not interrupted, and how to divide the memory contributed by these nodes to ensure the quality of service requirements of different priority tasks.

In this paper, experiments are designed to simulate user behavior and the effect of TM usage. In order to compare the effects of experiments, four systems are designed, including the original Linux system, the statically partitioned TM usage algorithm (divided by 20%, 40%, 60% for the three algorithms applied locally). The experimental results are shown in Table 1, divided into three groups, and the working set of local applications is continuously increasing. The TM usage algorithm consistently performs the best in all three cases, except that the algorithm with 60% allocation to the local node is slightly better than the TM usage algorithm when the working set is very small. Among the three sets of results, the original LINUX has the worst performance, with a maximum error rate of 164 page faults/sec. In general, this results in at least a 50% performance drop, which is obviously unacceptable to users.

Table 1. Page faults per second for TM algorithms with different working set sizes

	s	m	l
Linux	35	57	164
Allocate 20%	16	38	143
Allocate 40%	8	22	85
Allocate 60%	2	1	4
TM usage	3	0	6

Another feature of the TM algorithm is that the largest number of mining nodes contribute memory resources under the premise of ensuring application transparency. As shown in Table 2, the TM usage algorithm is the most robust in terms of the amount of memory contributed by nodes under different working set sizes. The robustness here does not refer to the stability of the amount of contributed memory, but should be viewed in conjunction with Table 1. The TM usage algorithm always makes the node contribute the best but not necessarily the most memory capacity, so that local applications are not affected.

Table 2. Memory contributed by TM algorithms with different working set sizes

	s	m	l
Allocate 20%	245	272	283
Allocate 40%	116	128	134
Allocate 60%	53	67	77
TM usage	169	81	42

4.2. Multi-Queue Node Scheduling

The multi-queue node scheduling algorithm ensures that distributed applications enjoy different relative service quality requirements according to different priorities in terms of CPU service time. During the design and implementation process of the algorithm, distributed applications are divided into multiple CPU waiting queues according to their different priorities, and the waiting queues with higher priority will be allocated more CPU service time. For distributed applications, the relative quality of service is directly reflected in the response time of tasks with different priorities. Distributed applications can only use the idle time slice of the CPU. Once the local application has a ready task, the system immediately interrupts the currently running distributed application and transfers the CPU usage right to the local application.

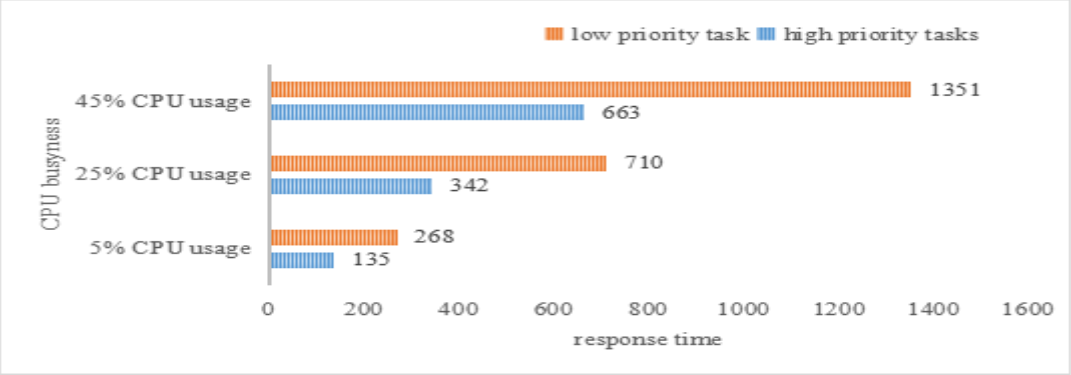

Figure 3. Effect of multi-queue node scheduling algorithm on relative quality of service

In the experiment, three groups of different local computing tasks are selected, which occupy 5%, 25%, and 45% of the CPU on average, respectively. The average response time data of tasks with different priorities of distributed applications are obtained as shown in Figure 3. It can be seen from the three sets of data that the response time of low-priority tasks is about twice that of high-priority tasks. The reason for this result is that in addition to the service time they obtain is about twice as long, there is also memory Unfair distribution also exacerbates this trend.

5. Conclusion

In this paper, a DS CM is constructed through the mobile Agent development platform, and the current DC problem is solved by using the characteristics of Agent running autonomously in a distributed environment. The functions of the system include a session connection module, a message transmission module, and an Agent communication module, and each module complements each other to complete the Agent computing task. This paper evaluates the performance of DC through the impact of TM algorithm and multi-queue node scheduling algorithm on service quality, and meets the user's service quality requirements for different tasks. It is hoped that the DC system in this paper can play its role in large-scale data processing. Help people improve data processing efficiency.

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] Fujikawa J, Shiokawa S. Information-Centric Architecture Using Mobile Agent for MANET.

Journal of Signal Processing, 2018, 22(4):179-183.

[2] Adegbite G, Emuoyibofarhe J, Ajala F A, et al. Development of a Multi-level Mobile Agent Security Model for Online Shopping Systems. International Journal of Wireless and Mobile Computing, 2019, 4(4):87-93.

[3] Salah-Ddine K, Laassiri J. Mobile Agent Security Based on Mutual Authentication and Elliptic Curve Cryptography. International Journal of Innovative Technology and Exploring Engineering, 2019, 08(12):2509-2517.

[4] Arora P K, Bhatia R. Mobile Agent Based Regression Test Case Generation using Model and Formal Specifications. Iet Software, 2018, 12(1):30-40.

[5] Chaudhary S, Kumar U, Gambhir M. Review and comparison of Mobile Agent Itinerary Planning Algorithms in WSN. International Journal Of Computer Sciences And Engineering, 2019, 7(6):209-219.

[6] Osero B O, Abade E, Mburu S. Mobile Agent Based Distributed Network Architecture with Map Reduce Programming Model. Computer Science and Information Technology, 2019, 7(5):129-161.

[7] Yadav S, Mohan R, Yadav P K. Fuzzy based task allocation technique in DC system. International Journal of Information Technology, 2019, 11(1):13-20.

[8] Saxena K, Abhyankar A R. Agent-Based DC for Power System State Estimation. IEEE Transactions on Smart Grid, 2020, PP(99):1-1.

[9] Zak M, Ware J A. Cloud based Distributed Denial of Service Alleviation System. Annals of Emerging Technologies in Computing, 2020, 4(1):44-53.

[10] Khandelwal A. Fuzzy based Amalgamated Technique for Optimal Service Time in DC System. International Journal of Recent Technology and Engineering, 2019, 8(3):6763-6768.

[11] Lenzen C, Patt-Shamir B, Peleg D. Distributed distance computation and routing with small messages. DC, 2019, 32(3):1-25.

[12] Firouzi R, Rahmani R, Kanter T. Federated Learning for Distributed Reasoning on Edge Computing. Procedia Computer Science, 2020, 184(6):419-427.

[13] Memon K A. Analyzing distributed denial of service attacks in cloud computing towards the Pakistan information technology industry. Indian Journal of Science and Technology, 2020, 13(29):2062-2072.

[14] Ketu S, Mishra P K, Agarwal S. Performance Analysis of DC Frameworks for Big Data Analytics: Hadoop Vs Spark. Computacion y Sistemas, 2020, 24(2):669–686.

[15] Samidurai R, Sriraman R, Zhu S. Leakage delay-dependent stability analysis for complex-valued neural networks with discrete and distributed time-varying delays. Neurocomputing, 2019, 338(APR.21):262-273.

[16] Jarraya A, Bouzeghoub A, Borgi A, et al. DCR: A new distributed model for human activity recognition in smart homes. Expert Systems with Application, 2020, 140(Feb.):112849.1-112849.19.

[17] Froelicher D, Troncoso-Pastoriza J R, Sousa J S, et al. Drynx: Decentralized, Secure, Verifiable System for Statistical Queries andMachine Learning on Distributed Datasets. IEEE Transactions on Information Forensics and Security, 2020, PP(99):1-1.

[18] Malaviya A. DCDedupe: selective deduplication and delta compression with effective routing for distributed storage. Computing reviews, 2019, 60(2):75-75.

[19] Hakeem A, Curtmola R, Ding X, et al. DFPS: A Distributed Mobile System for Free Parking Assignment. IEEE Transactions on Mobile Computing, 2020, PP(99):1-1.