Using Ant Colony Algorithm on Scheduling Strategy Based on Docker Cloud Platform

Guanqaun Wu
Dongguan University of Technology - City College
Rongli Chen (23458920@qq.com)
Dongguan Polytechnic
Desheng Zen
Guangdong Innovative Technical College
Xiaozhong Chen
Dongguan Polytechnic

Research Article

Keywords: Artificial intelligence, Ant colony algorithm, Cloud platform, Docker cluster, Task scheduling

Posted Date: June 18th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-619916/v1

Using ant colony algorithm on scheduling strategy based on docker cloud platform

Guanqaun Wu[1], Rongli Chen[2*], Desheng Zen[3] ,Xiaozhong Chen[4]

[1]Department of Academic Affairs Office, Dongguan City University, Dongguan, 523419, China,Mail:623948587@qq.com
[2] Department of Computer Engineering, Dongguan Polytechnic, Dongguan, 523808 China.Mail: 23458920@qq.com
[3]School of Information Engineering,Guangdong Innovative Technical College, Dongguan, 523960, China. Mail: 47328625@qq.com
[4]Department of Logistics Engineering, Dongguan Polytechnic, Dongguan, 523808, China，Mail:9746042@qq.com
*Corresponding author: Rongli Chen (23458920@qq.com)

Abstract: With the development of Internet technology, the combination of industrial intelligence and data mining technology has played a huge role in the development of various industries, and has also become an inevitable demand for the development of information society. The cloud service can solve the problem of data processing conveniently and quickly, so it becomes particularly important for the development of cloud service. Traditional cloud service applications cannot meet users' urgent needs for services in terms of rapid release and efficient allocation of resources. Container technology has emerged to meet this challenge. Containers as a better resource virtualization technology will naturally be widely recognized and favored by the industry. Container cloud platform has the advantages of small redundant environment, high resource utilization, fast deployment, flexible and scalable. The proposed method use task scheduling of container cloud platform is to allocate available resources to users according to their different service demands. The result of this paper can ensure the effective execution of user container tasks.

Keywords: Artificial intelligence; Ant colony algorithm; Cloud platform; Docker cluster; Task scheduling.

1. Introduction

The main content of this study is to existing Docker native Swarm cluster problems of scheduling policy is optimized, the focus is on the container cluster resource scheduling in task distribution load uneven distribution of resources, the cluster of the poor overall performance, using heuristic intelligent algorithm of ant colony algorithm for scheduling optimization to improve, due to the disadvantages of ant colony algorithm, first of all disadvantages to improve the algorithm, and then apply the improved algorithm Swarm system structure model design, accomplish more detailed reasonable allocation of resources. The main contents of this paper mainly include the following four points:

- The implementation principle of relevant Docker technologies is analyzed to prepare for subsequent improvement.
- The original working principles of Kubernetes, Mesos and Docker are

studied and analyzed, with the focus on the summary of the principles, advantages and disadvantages of Swarm.

- To summarize the shortcomings and shortcomings of heuristic ant colony algorithm, the scheduling policy that the minimum task is completed first is used to initialize pheromone to reduce the starting time of search. Secondly by introducing balance factor, to coordinate the local and global pheromone pheromone update mechanism, the ant colony can according to the current iteration load and pheromone update mechanism to determine the next round of strategy, at the same time also introduced volatilization coefficient adjustment mechanism, to improve the global search ability of the algorithm, making optimal use of resources of each node.
- An improved Ant Colony Optimization (ACO) algorithm is proposed to optimize cluster resource allocation.Aiming at the existing shortcomings of ant colony algorithm, the scheduling policy that the minimum task is completed first is used to initialize the pheromone to reduce the starting time of search. Secondly by introducing balance factor, to coordinate the local and global pheromone pheromone update mechanism, the ant colony can according to the current iteration load and pheromone update mechanism to determine the next round of strategy, at the same time also introduced volatilization coefficient adjustment mechanism, to improve the global search ability of the algorithm, making optimal use of resources of each node(Ma et al.,2015).

2. Relative work

2.1 Cloud computing concept

Cloud computing is a computing model.Users can access shared computing resource pools on demand, efficiently and conveniently through the network, and these resources can be quickly provided to users by minimizing administrative costs or interacting with service provision. In simple terms, cloud computing vendors build pools of resources by pooling large amounts of cloud infrastructure, then use virtualization to isolate the resources and rent and sell them to users.

2.2 Container Technology

Container is a new virtualization technology. It balances resource usage conflicts by dividing resources managed by the operating system into separate groups(Yang et al.,2015). Compared with virtual machine technology, container technology is less isolated than virtual machine technology, but the container is significantly faster in startup and running speed than virtual machine technology. The advent of container technology has revolutionized the way people develop, distribute, and run software. By using containers, software developers do not need to care about the actual environment in which the software is running. Operations personnel can also greatly reduce the workload of configuring the environment and system dependencies, and only need to focus on network and resource maintenance to ensure the uptime of the software. At present, container technology has been widely used. Gartner research report (Petrucci et al.,2015).It notes that around 70% of the world's applications are currently deployed to run on container infrastructure. As a result, containers are

becoming a very important piece of infrastructure in cloud data centers.2.1.3 Contrastions between Containers and Traditional Virtualization Containers and virtual machines are very different in the underlying implementation. Container technology is virtualization at the operating system level, while traditional virtual machines implement virtualization at the hardware level based on Hypervisor (Zhan et al., 2003). The architectural differences between virtual machines and containers are shown in Figure 2.1. Structurally, the container does not have a Guest OS layer and a Hypervisor layer. The container shares the operating system with the host, thus saving some of the process of traditional deployment applications, such as configuring the environment, resolving dependencies, etc. Containers are lighter than virtual machines and are typically MB. At the same time, containers are much closer to bare metal in terms of performance than virtual machines.

Since Docker container shares the host's operating system, network, file system and process can be isolated through the namespace namespace of the kernel. Meanwhile, system resources can also be shared between different Docker containers. Therefore, Docker container consumes less resources and is more lightweight compared with virtual machines. There is almost no resource sharing between different virtual machines, so virtual machines are more resource-intensive and heavier than containers. Containers can start up in seconds or even milliseconds, whereas virtual machines typically start up in seconds or tens of seconds. Typically, thousands of containers can be deployed on a single physical node, but typically only a few dozen virtual machines can be deployed.In addition, container images take up only a small portion of disk space, typically at the MB level, while virtual machine images may be at the GB level.A comparison between container technology and traditional virtual machines is shown in Figure 2.1.

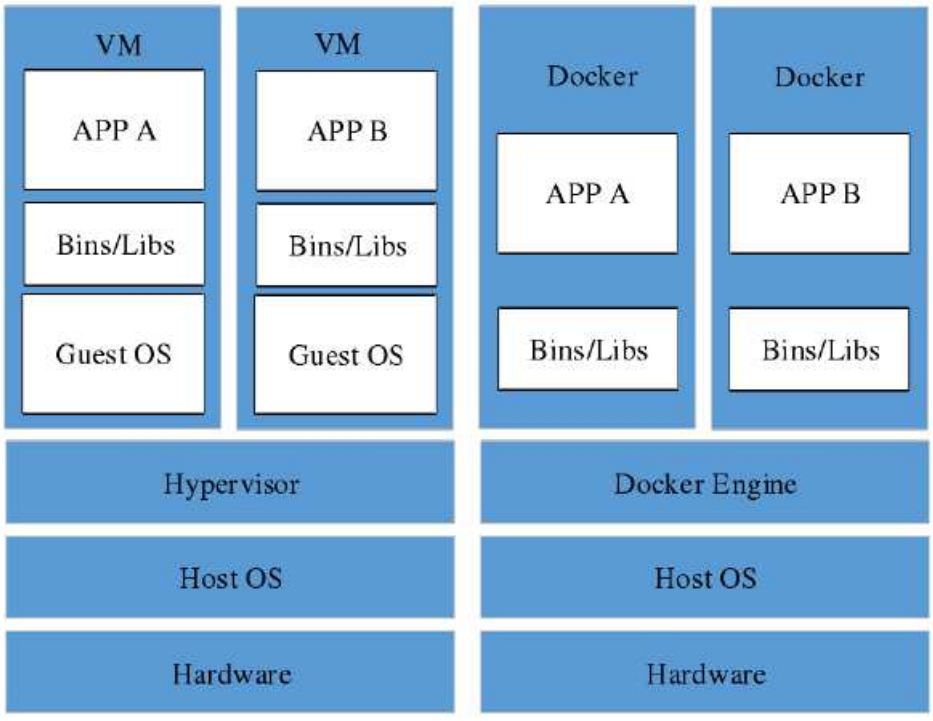

Figure 2.1 Comparison of a traditional virtual machine (VM) and a Docker

Container

The emergence of Docker technology is undoubtedly a revolution in the field of cloud computing. It has established a system and ecology for the construction, distribution and execution of a set of applications with container technology as the core, which is the Docker ecology in its heyday(Li et al., 2019). It must have its unique advantages.The technical advantages of Docker are described in Table 2.1 below.

Table 2.1 Advantages of Docker Container Technology

advantage	role
1. Continuous deployment and testing	Container technology eliminates the difference between online and offline, ensuring consistency and standardization of the application environment. Test and operation personnel can directly deploy the developer's packaged image containing code, environment, dependencies, etc., which is convenient and quick, simplifying the deployment and testing process.
2. Versioning of standardized environments	Compared with Git for code versioning, Docker supports not only code versioning, but also the implementation of version control in conjunction with the application runtime's standardized environment. Once an error can be easily and quickly rolled back.
3. High resource utilization and resource isolation	Docker directly interacts with the underlying operating system and does not require any management software, which naturally eliminates unnecessary system load and overhead. Multiple container instances can be started under the same conditions to make full use of system resources. At the same time, Docker's own isolation mechanism can accurately limit the container resources.
4. Cross-platform and mirroring	Relying on the included application machine standardized runtime environment image, Docker container can be executed on any platform that supports Docker, and does not need to do any processing and adjustment, deployment is easy and fast, truly achieve "one build, multiple runs".
5. Easy to understand	"Container", vividly likens the container to the container itself, and the applications and their runtime standard environment inside the container are the goods inside the container. The isolation and portability of Docker are well reflected.
6. Apply mirror repositories	Docker officially provides a mirror repository, which is organized and managed in a similar way to Git Hub. Docker itself has cross-platform and portability, which is equivalent to providing users with an all-inclusive application store, which provides great convenience for developers.

2.3 Key technologies of Docker container

As is known to all, the virtual machine is by running the virtual machine on the

host management program simulation real physical machine system to implement virtualization, based on the virtual machine environment system of limitation and isolation can realize resource encapsulation, but when the virtual machine and host machine system, the application of the virtual machine can not use host resources directly, so often a waste of system resources.By constantly refining the namespace[1](Namespace)(Liu et al.,2015). Developers have solved the above problems by allowing some processes to share the same kernel in an isolated Namespace. This is like sharing a bedroom, where tenants share bathrooms and living rooms, but each tenant is isolated from each other. In contrast, apartment suites are equivalent to virtual machines, and the sharing mechanism is similar to container technology. Tenants are equivalent to containers. When using system resources, each container thinks that it is unique and monopolizes the resources.Namespace is used to isolate the running environment, and Cgroups are used to control and allocate host system resources(Cai et al., 2017).

1. Namespace

Container technology uses namespaces as a way of isolating containers so that applications running in containers can be executed as if they were in the operating system. Namesapce typically includes six types, as shown in Table 2.2. Because container technology is created in a namespace based on the operating system kernel, it makes less resource usage than virtual machines(Yuan et al., 2019) .

Table 2.2 Namespace types

Namespace	Isolate the content
UTC	Host name, domain name
IPC	Semaphore, message queue
PID	Process of no.
Network	The network information
Mount	mount
User	User groups

2. CGroups

Container technology adopts Cgroups technology. Cgroups technology can control CPU, memory and other physical resources in a more fine-grained way to limit the resource usage of each process, and users can use the applied resources in a more flexible way.

3. Docker file technology

Docker designs a layered mirrored file format. This file format is composed of the uppermost read-write layer combined with multiple read-only layers. By federating the file system mount to be a container-only file system, when the user does file operations, all operations are kept in the read-write layer, not the read-only layer mirror is written again.When users need to customize their own image, they can directly submit their own image on the basis of the original image, or write it with Dockerfile on top of it. This hierarchical file storage method facilitates the reuse and expansion of the image, reduces the redundancy of the file when iteration, and ensures the stability of the whole system.

Docker is the equivalent of a program that runs on the host computer. This program is a process (Liu G et al., 2017), it uses Namespace to isolate resources, uses control group CGroups to limit the use of resources, and uses efficient copy-on-write mechanism to isolate processes between programs currently running in the Docker container and those in the host. If you want to know more about the current process structure, you can use the command to display it on the host. Figure 2.2 shows the process tree structure of Docker container. The three processes associated with Docker containerd are init, dockerd and Docker containerd, and shim and /bin/bash represent different processes. Its implementation can create a process through the clone command in the Namespace to realize the isolation of its process, so that the process in the host and the process in the Docker container do not know each other and deal with their own affairs respectively.

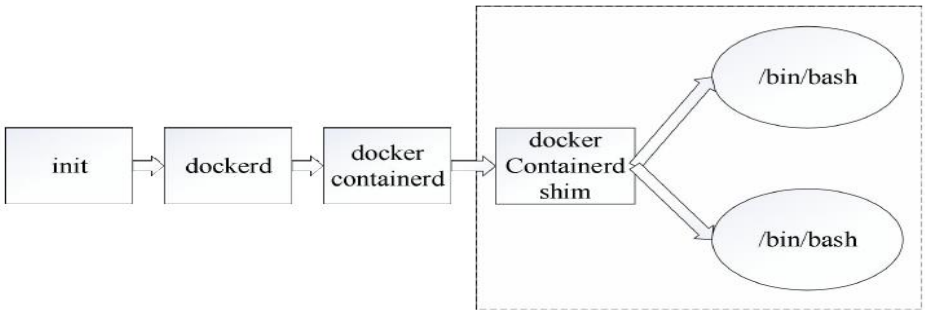

Figure 2.2 Process tree of Docker container

2.4 the Docker architecture

Docker adopts C/S(Client/Server) mode. Figure 2.3 shows the architecture model of Docker(Lv et al., 2019), the user can send the request task to the back end through the Client, and the back end Daemon will reply after receiving the request. In this way, the front-background interaction is realized. The general method to handle the request is some operations on the container and the mirror.

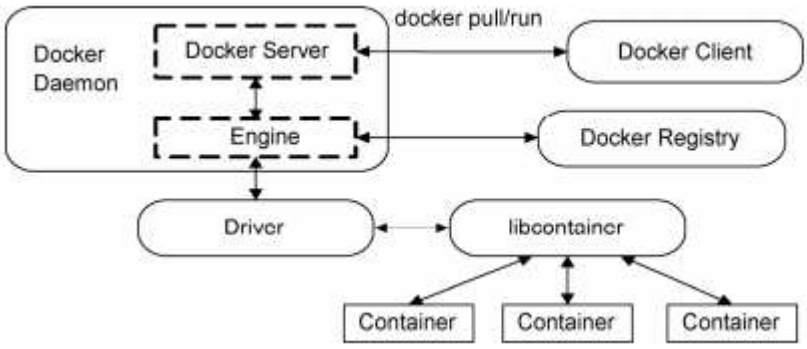

Figure 2.3 Docker System Structure Diagram

The client/server mode used by Docker can be viewed by using the command "dockerversion" on a system with Docker.Client users create, manage and use containers through the use of remote API interfaces.

Docker Daemon is composed of Docker Server and Docker Engine.Docker

Engine is responsible for the specific execution operations. Docker Driver plays a driving role and is responsible for the layout of the dependent environment for container operation. LibContainer is a dependent library, the bottom layer is written in GO language, and Docker Registry is the image warehouse of Docker.An image can be an application, system files, and their dependencies that are published in the repository and can be used directly locally.

Docker background daemon is mainly composed of Docker, Engine, and the Job Server, the Server is Docker background services between, its various requests received from the client, and according to the request command format of different tasks allocated to different container Engine interface, implement the related methods, its implementation through the Docker ServerAPI call execution units.

Docker Engine is equivalent to the brain of the background service process. It is a key part of the background request processing. It contains some contents of the container description and controls most of the execution units. The Engine assists with most of the background tasks and calls related execution units through it. In addition to these basic background services, it also handles process exceptions.The unit of work mentioned above is the Job in the engine. How many jobs will the Docker background execute the request of the front end? When the Job is completed, it means the end of a task(Liu G et al., 2017).

2.5 Docker mirror technology management

A software system has both the system itself and the parts provided to the user. When a Linux system is started, it provides services to its users by associating root file system operations, and Docker's implementation is no exception, which is mirrored by Docker(Lu et al., 2018), similar to the file system, the root file system is a series of file resources that the container normally opens and works on, and the file system resource is unwritable. The Docker image is all static data. In other words, when a mirror is created, the information in it cannot be changed.

Docker image related technology There are four: mirror layering, copy-on-write, content addressing, and federated mount(Liu M et al., 2018).

Docker uses the Image layering technology to solve the problem that Image occupies a large amount of host storage resources. Its principle is equivalent to Docker containers between host and the process of resources to solve the problem, when more than one image content in phase at the same time, you can use a same content used as multiple receptacles, thus reducing the mirror warehouse storage resources, at the same time, this mechanism can greatly reduce consumption of resources for the management of the mirror, reducing the pressure on the host hardware resources, improve the operation efficiency. By layering the common content in the image, Docker's image is made more lightweight. If we want to change the mirror content, we only need to change the most superficial layer in the multi-layer, and the original one will still be on the disk but will not be used anymore. For example, when a change is made to a Web-related service image that uses this technique, the change can only be made by creating a new layer that stores the run-dependent environment and associated run-support, which has only read permissions and cannot be changed. On this layer, the read-write operation can be

performed by creating a new layer.Docker image uses copy-on-write mechanism to realize the sharing of the original mirror resource, cancells the copy of the original mirror file, and uses read permission to re-associate the mount point.Based on mount implementation on the read - write permissions for a layer, such as file does not change when the container is used by the same set of content, if the container when use file system changes, will all change the file content to the top the read - write access layer, and the original cover the read access layer content, equivalent to subclasses in the object oriented to rewrite the superclass method, when using this call this method will only choose the subclasses override method, here the parent class is equivalent to a readable layer,Any other subclass can override its inherited base class methods.

Content-Addressable Storage (Content-Addressable Storage) is a major modification of Docker image in the new version of Docker. The principle of this mechanism is to find the image and its level according to the file data.Before is to adopt the principle of the image layer produced by different random code to represent the image layer, and the principle of the image data after changes the checksum calculation, using the checksum hash operation going on, to obtain a value to replace the original random code, using the obvious advantage of the hash value, on the one hand, to strengthen the reliability of the mirror, in execution of mirror image acquisition, for example, upload, save and load after the command, can undertake judgment of image data is complete, on the other hand, using the hash value of relative random code to reduce the emergence of conflict.If two mirrored files have the same hash value, you can mirror a share between them even if they are not part of the same build.

3 Cluster Container Choreographer and Docker Swarm

Docker technology has the very high value, and promote the development of the entire Internet ecosystem, greatly improve the application development, deployment and operation efficiency, and production in the actual operation environment, and faced with many problems, such as multiple containers of container under the network environment of cloud computing and storage resources allocation, and the management of the cluster, load balance, reliability problem to be solved, this has given rise to the container cluster management tool, so that users can use a layout tool instead of using the API direct management control container.

3.1 Kubernetes

Kubernetes Google is an open source container orchestration engine, often referred to as K8S, used for container cluster scheduling management. Its distributed mode provides a method of rapid deployment, management and maintenance of containerized applications(Ling et al., 2018).

3.1.1 Kubernetes architecture

Kubernets is a container orchestration tool developed by Google based on the Go language(Liu G et al., 2017). Like Swarm, Kubernets is an open source containerized application for managing multiple hosts on a cloud platform.With automatic deployment, application container management and elastic scalability and other functions.The architecture of Kubernets is shown in Figure 3.1.

The container group Pod is at the heart of Kubernets. POD contains a set of containers and volumes. Containers in a Pod share a network namespace and can communicate with each other using localhost.

With KuberNets, users can manage, control, and monitor the deployment of containerized applications across a cluster of containers across hosts. Kubernets is an extensible, portable container management framework. Companies like IBM, Microsoft, RedHat, Docker, SaltStack, CoreOS, and Mesosphere have joined the Kubernets community. Kubernets use a Master/Slave structure, where a Master Node in a cluster can manage containers on multiple Node nodes. The main steps of scheduling are as follows:

Step 1: Filter out all resource nodes that do not meet the requirements.

Step 2: Score the Node by calculating the CPU utilization and memory utilization of each Node in combination with the scheduling algorithm, and the container will be deployed on the Node with the highest score.

Thus, the scheduling algorithm of Kubernetes has some limitations(Liu M et al., 2018) . First, Kubernetes only considers CPU and memory when allocating resources. Network bandwidth, disk I/O and other types of resources also have a significant impact on application performance.

Secondly, the SPREAD algorithm is the default scheduling algorithm in Kubernetes. The algorithm distributes the workload as evenly as possible across the nodes.This will lead to insufficient utilization of resources on each node, and may also lead to excessive energy consumption of the cluster.

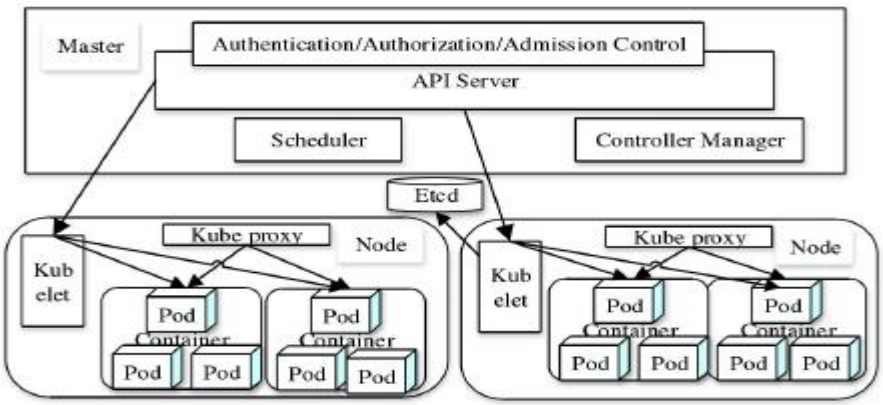

Figure 3.1 The architecture of Kubernetes

3.2 MESOS architecture and scheduling principles

Mesos is an open source distributed resource management framework under Apache that supports large-scale scaling of container clusters. MESOS provides efficient isolation and sharing of resources across distributed applications and frameworks, supporting Hadoop, MPI, Hypertable, Spark, and more.

The architecture of MESOS is shown in Figure 3.2.Mesos manages slave daemons on resource nodes through master daemon, and Frameworks runs tasks on Slaves.

Figure 3.2 describes the Mesos related components, it can be seen that Mesos is a typical Master/Agent architecture [29], the Master is the same as the Master of

Kubernetes system is the management node, Mesos Master general set to an odd number in the system and is connected between the Mesos and through a mechanism for elected leader, elected leader is responsible for providing services to outside,The other masters will act as backups to the leader. It is possible to add different computing frameworks to the Master, and these computing frameworks can call the Master API to create request tasks. Each Master is composed of multiple computing nodes, each compute node has a Master Agent, the Agent shall be responsible for the nodes on the computing resources information to the Master, the Master will be issued to all the requests for computing nodes, then the task as a container to run, while the Mesos Agent Containerizer containers in the engine will be used for the container to create, update, manage, monitor, and destroyed.

Key components and concepts: ZooKeeper is mainly used to implement Master election and support Master high availability. The Master Mesos Master node accepts the Slave and Framework Scheduler registration and allocates resources. From the node, the Slave receives a task from the master and dispatches the executor to execute it. Framework includes Scheduler and Executor. When the Scheduler is started, it registers with the master and accepts the Resource Offer sent by the master to decide whether to accept or not. Executor is called to slave to execute framework tasks. Mesos has CommandExecutor and DockerExecutor built-in, and other custom Executors need to provide URIs for slaves to download.

The main Task of the Task Mesos is to allocate resources and then ask the Scheduler if he can use them to execute tasks. The Scheduler then binds the resources to the Task and sends them to the Master to execute the Salve. Tasks can be long lifetime or short lifetime in batch.

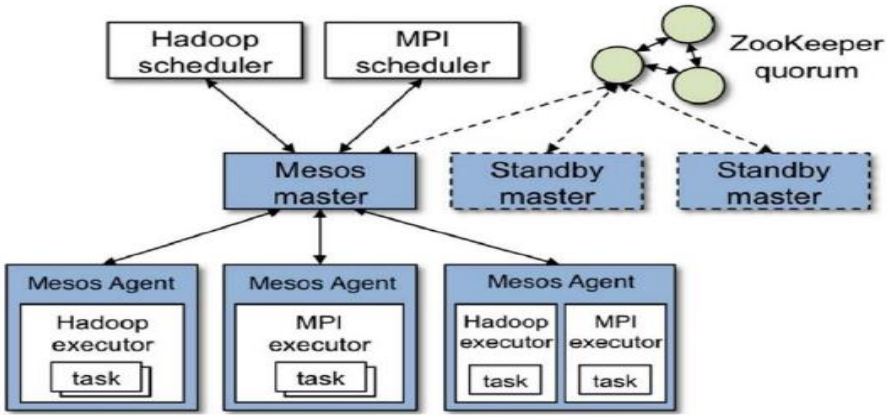

Figure 3.2 MESOS architecture diagram

The MESOS architecture is designed to efficiently share hardware resources among different frameworks, while simplifying its scheduling logic and making it as compatible and extensible as possible, so as to ensure its robustness in a large-scale clustered environment and its universal applicability to all possible computing frameworks.The default MESOS resource allocation strategy is the DRF algorithm. The goal of the DRF algorithm is to ensure that each Framework receives a fair share of its most needed resources in a heterogeneous environment.

3.3 Docker Swarm

Docker Swarm is Docker's native cluster scheduling and choreography tool(Medel et al., 2018,Zhang et al., 2014)Docker Swarm is used for container cluster management for developers. Using Docker Swarm for container management can make users feel as if they are operating a single container. In fact, this single container is the result of the collaborative work of background container cluster.The management tools to realize the resource abstraction of physical hosts, but also realized the container cluster scheduling allocation and management of resources, set utility at a suit, in solving the problems such as rapid deployment and extension services on different host obvious advantages, the cluster management plan as a whole to the container vessel engine, make do not need to use other software tools to other cluster management.

Swarm has many advantages. First of all, as a native cluster management tool of Docker, it is updated and maintained by Docker project developers and open source community related developers. It can directly use Docker related APIs without the support of other software, and has unique advantages in software compatibility. It makes Swarm easy to integrate with Docker related software, Docker Compose, Docker-py, etc., which is convenient for system porting and greatly improves the efficiency of project deployment. Moreover, Swarm provides internal network pluggable interface support, so that developers can easily conduct cross-host container interaction to provide services.

Secondly, Swarm is often used in the real Internet environment. With its strong background and open source community, Swarm continuously improves and improves the container management, so that it can support the management of clusters composed of thousands of containers.In addition, when users need to use Docker Swarm, there is no need to download and install additional software. Swarm is embedded in Docker software, so users can use it more conveniently. Docker Swarm relies on its strong open source community, and the project is still alive and well, with many developers participating.

Due to early Swarm API [33] many of the same as the Docker API interface, and its function is different also, therefore caused a lot of trouble to the developer, Swarm API in order to realize the interface functions and some unique functions such as cluster management, information on cluster view and log monitor prominent problems such as the container cluster, for example in the early days of the Docker command window input "Docker info" command, it will only show some Docker engine related information,However, typing this command in Swarm should show information about the container cluster.

At the 2016 Docker Developer Conference, Swarmkit, a new marshalling tool based on Docker Swarm (used in this discussion), was unveiled to run large distributed systems on it. In Swarkit, the old scheduling methods of Spread and Binpack are still included, while some filters and service discovery mechanisms are still in place. These filters filter out illegal nodes and check the integrity of the software, such as runtime dependencies, ports and security. Swarm uses proxies to register its discovery service by storing key-value pairs. The next section examines its architectural model in detail(Yuan et al., 2019) .

Docker has integrated Swarm into the Docker engine. Make Swarm the built-in cluster management system. If you want to create a container cluster, you can use the command "docker cli" to do so. Then you can deploy the application, which will eliminate the need for third party software to create and manage the cluster.The important features of Swarm clustering are as follows(Yuan et al., 2019,Lv et al., 2019).

1. By using the management node and node work mechanism, has the characteristics of decentralization, can be done by the Docker engine for all types of nodes deployment, in addition, when the cluster system is running, you can control through the Swarm clusters all the changes, such as the cluster size larger or smaller, increase or decrease the management node and work nodes, etc., and the cluster of these changes can be achieved when system is running, is very convenient and practical.

2.Docker engine adopts the way of declaring service, which can be used to facilitate users to define the service state they want to see in the application stack. If we create a new Web service stack, it includes front-end business, a back-end business and database business, and the front-end business depends on message queue business to realize.

3. When run on an application to deploy, you can manually on the command line, to set up how many resources are needed to run it or how many containers, open for online applications running, if due to business needs, the need for expansion, also can be set from the command line to increase the running on the number of containers, makes the Swarm is more convenient and efficient expansion.

4. Management of Swarm clusters nodes to real-time monitor the running status of the entire cluster, if cluster appeared some unusual Swarm will adopt automatic repair function, to realize the system normal operation, such as when an application needs to start open 12 container to run, work nodes running container has two can't work normally, this is Swarm management node will redistribute the other available work nodes to replace the node does not work.In this way, the normal operation of application service can be realized through cluster monitoring and automatic repair.

5.Swarm can connect to multi-host networks. It uses Swarm management nodes to provide an Overlay network for a cross-host container cluster to assign a virtual IP to the container.

6. The nodes in Swarm use two-way authentication and communicate with encryption on each node and between each node in the cluster.

3.3 Swarm architecture, service discovery and scheduling strategies

Swarm is different from Mesos, which requires the installation of Docker, Marathon, and ZooKeeper. The operation of Swarm is relatively simple. As shown in Figure 3.3, if we have 3 LiunX servers, each of which is equipped with Docker, select one as the manager and execute the first command:Docker Swarm Init -- listen-addr will print a token as Docker Swarm credentials after execution. Then execute the second command under each worker node: Docker Swarm Join-Token < Token >< manager-IP >:< Port >, which means to join the cluster, all you need is the token and the IP and PORT number of the corresponding MANAGER node, and the cluster

environment will be completed.

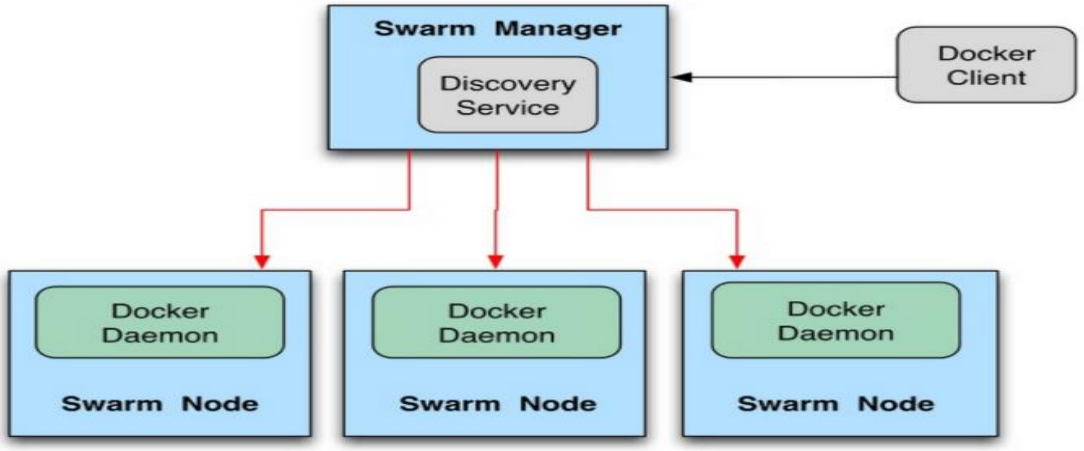

Figure 3.3 Swarm cluster setup

In the Swarm, is composed of two kinds of nodes, cluster management node and node, a node is composed of the container cluster instance, when users remotely task request, the request information will first through the cluster management node, then the management node will work for each node and the allocation of resources to tasks and execution. To avoid cluster "brain-splitting" failures, the management nodes are set to an odd number.

Service discovery is a bit more complex and is divided into three main categories, Ingress, Ingress+Link and Custom, depending on the scenario. While Ingress is based on a virtual network on top of a physical network, Swarm's upper-level applications do not rely on the physical network and can leave the underlying physical network unchanged. I won't go into details here.

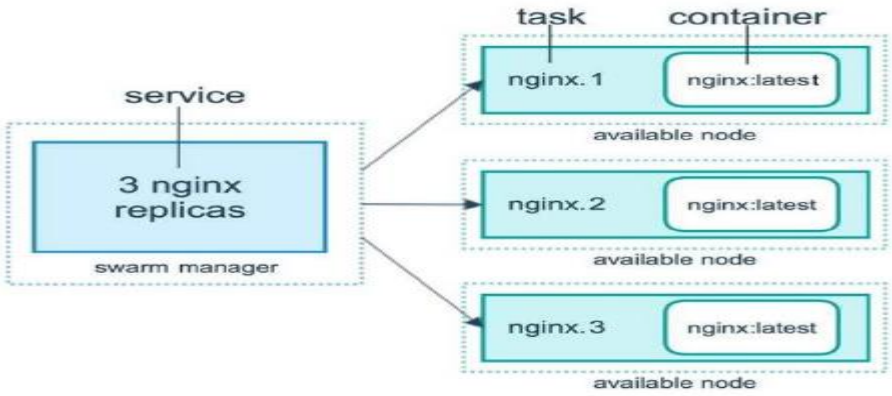

Figure 3.4 Service, Task, Container Diagram

Figure 3.4 illustrates the relationship between three key concepts in Swarm cluster scheduling, namely service, task, and container, by providing NGINX services.

From a developer's point of view, the task scheduling module Implementation of don't need to know how the underlying implementation and rely on the platform environment, the focus is on how to assign task reasonably provide resource node in the cluster, first of all, the management node will place each node can provide resources (CPU, RAM, etc.) information to carry on the summary, provided a basis for the follow-up to the allocation of resources, providing resources in this process

and Mesos scheduling phase (Offer), Swarm called resource abstraction stage, when I received the task request, will obtain the allocation of resources from the collection of information resources information(Docker et al.,2016).

After managing node resource information acquisition, the next process is Service. A Service is composed of one or more processes, and each process needs to be allocated resources to run in the work node to ensure the normal operation of the Service.Swarm refers to each process in the service as a Task. This Task is the content that runs in the container. This Task uses the resources requested when the service is started.

In Swarm, there are two ways for clients to initiate docker commands: one is to SSH directly to the manager node and execute docker commands.The other option is to call the docker command on the Manager through the Remote API. The second option is shown in the following diagram.

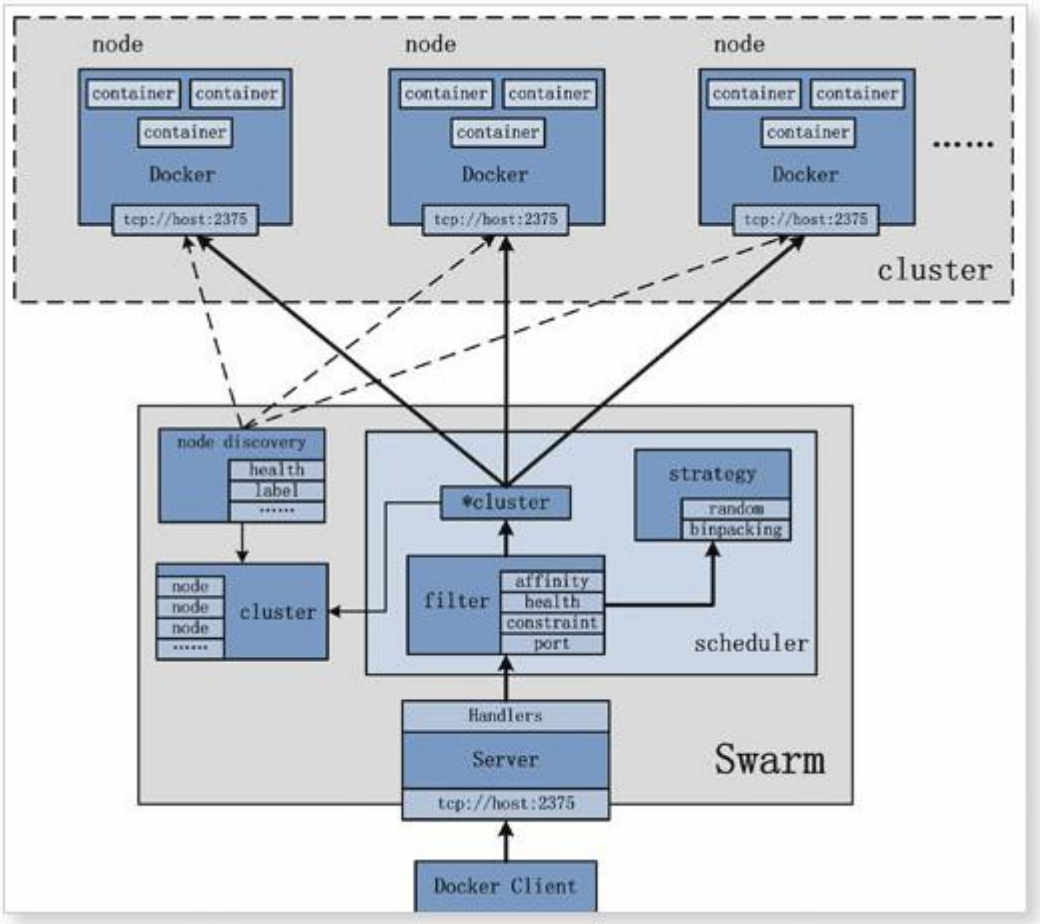

Figure 3.5 Docker Swarm structure diagram

Docker Client outside the manager node, if executed the Docker service create, first passes through Docker Daemon to accept this command, to the Scheduler module, the Scheduler module mainly realize the scheduling function, select the most appropriate nodes, it has two parts, Fiter and Strategy, Fiter is used to filter node, satisfy certain conditions is obtained by filtering nodes (normal) resources enough, node,Strategy is to filter out after selecting the optimal node (compared to select

resources most remaining nodes, or find resources minimum remaining nodes), Fiter and Strategy are, of course, the user can separate custom, in the middle of the Cluster is abstract worker node Cluster, contained the Swarm nodes of each node information, on the left side of the Discovery is information maintenance module, such as the Label to the Health. The Cluster finally calls the API of the container to complete the container startup.

When creating a service, users will select the most appropriate node, and the selection process is divided into two stages. It first goes through the filter and then the scheduling policy is selected. Filter has four filters. Constraints are the Constraints that restrict the type of host operating system, kernel version, and storage that you are running on to the current conditions of the service application. The Constraints also allow you to limit the Constraints based on the developer's requirements. When the system is up and running, it can be filtered by using the LABLE tag to set some of the required specific attributes on the running system, so it is possible to get a more appropriate scheduling node. Affinity is a kind of Affinity filters, Affinity is the embodiment of the relevance, the filter is compatible to mirror and containers of Affinity, respectively, for example, a service, we put the database container and Web container in a piece, can use Affinity filter implementation, the Dependency is dependent on filters, link, etc., for example when create a container need another container support, the Dependency these containers will be on the same node. Health filter is a Health filter, which filters whether each node can work normally, and can judge the existence of faults in some nodes and filter out the nodes with faults. The Ports filter filters whether or not certain port numbers are used on a host node. For example, if a port 8080 is used on a host, and some hosts are not, then the hosts that are not used are selected. There are three methods in Strategy, which are Binpack, Spread and Random.

Swarm can join thousands of such a node in the cluster, and more of the problem is how to play the role of good resources, how to deal with the allocation of resources, reasonable and efficient use of resources to provide reliable service, if the container scheduling make improper use of resources, on the one hand provide hardware cost causes too much waste of resources, on the other hand, too little resources will affect the reliability of the service.The following is an analysis of each scheduling mechanism of Swarm's three strategies(Yuan et al., 2019;Medel et al., 2018):

1. Random scheduling mechanism.Random is a scheduling policy that existed in earlier versions of Swarm. It randomly selected nodes to deploy containers. This scheduling is obviously problematic and is only used during development testing.

2. Spread scheduling mechanism. Spread is the later version to join the scheduling policy, it will choose the resources such as CPU, memory, takes up the least amount of node priority deployment container, doing so will help keep the container is relatively evenly distributed on each node, ensure that all the nodes in the cluster resources used by uniform, but also a waste of resources, each host node resources utilization rate is low.

Binpack scheduling mechanism, Binpack is an early version of a scheduling strategy, its purpose is to fill up a node, as much as possible to the other node

deployment container, such can fully use each node resources as much as possible, save is not using node resources guarantee more resources to spare, but the host life shorter, more don't talk about load balancing.

The above three built-in resource scheduling strategies can meet some of the low efficiency requirements of the cluster applications. However, for some applications that need to make full use of the overall performance of the cluster and require high load balancing and resource utilization, the three built-in scheduling strategies obviously have great deficiencies. Among the three strategies, the spread strategy is generally used more. When using the SPREAD strategy, Docker□Swarm mainly allocates tasks according to the hardware conditions of each node, such as memory resources, CPU resources, etc. But Docker□Swarm doesn't take into account the actual resources available on each node. For example, when DockerSwarm determines that a node has sufficient memory resources as the optimal choice, it does so based on the amount of memory resources that the node has, and does not take into account the fact that the node's memory resources are already being used. If the node is running a memory intensive task, creating a new task on the node is not an optimal option. □ □

4. Application of ACO Improved Algorithm in Docker Swarm Cluster

This paper introduces and compares the architecture and scheduling strategies of container cluster arrangement tools Kubernetes, Mesos and Docker Swarm, and determines that the research focus of this paper is the problem of Swarm scheduling with uneven load and poor cluster performance when resource utilization fails to reach the expected effect.This chapter begins with the analysis of Swarm's working mode(Nguyen et al., 2017;Li et al., 2018), the ant colony scheduling algorithm was combined to improve the scheduling strategy, the scheduling deconstruction model was established, and the relevant experimental analysis was carried out.

4.1 Establishment of structural model

The implementation principle of Docker is equivalent to a container to put the application program and its dependent environment inside. This container can run its internal application on any host equipped with Docker software, realizing the rapid and efficient deployment and use of the application. However, it is also essential for developers to have a deep understanding of Docker's execution mode. Docker executes commands in two modes: Client mode and Docker Daemon.

When the user inputs relevant task request commands on the Docker command line, the background service daemon will receive them. In this way, after the connection between the front and rear ends is established, the background Docker daemon will parse the commands sent and then schedule the tasks and allocate resources to execute the tasks.A detailed analysis is given below.

After initializing the front and rear end startup of Docker system, the task process steps are as follows:

First the client sends the request from the task. After executing the command "Docker Run", the client will send the command information of the client to the server. At this time, the Docker client is in the client mode and will suggest a client and conduct initialization operation on it. Then, it will search for relevant methods to

execute the command through mapping, and the method here is generally run CMD.It is used to parse the commands entered by the user. After parsing, two HTTP requests are made, and the back-end is accepted and processed by the Server-related API.

Then the back end receives the request from the front end and creates the container. The newly created container object is created based on the form parameters passed by the front end.These container objects are the Go language related code "container_unix.go".After the container object is established, it responds to the previous segment, and the client sends a startup request.

Finally, you start the container. After the Server in Docker daemon receives the start request, it will call the relevant API to start the container, which runs the related program code of "start.go", so that the container will start on the host. At this stage, execDriver will interact with the host system, and it implements the namespaces and Cgroups functions internally.Based on the detailed analysis of the structure and scheduling of SWARM in Chapter 3, this paper proposes an ant colony algorithm to improve the scheduling of SWARM, aiming at the existing problems of three scheduling.

Figure 4.1 shows the proposed Swarm based ACO scheduling structure.

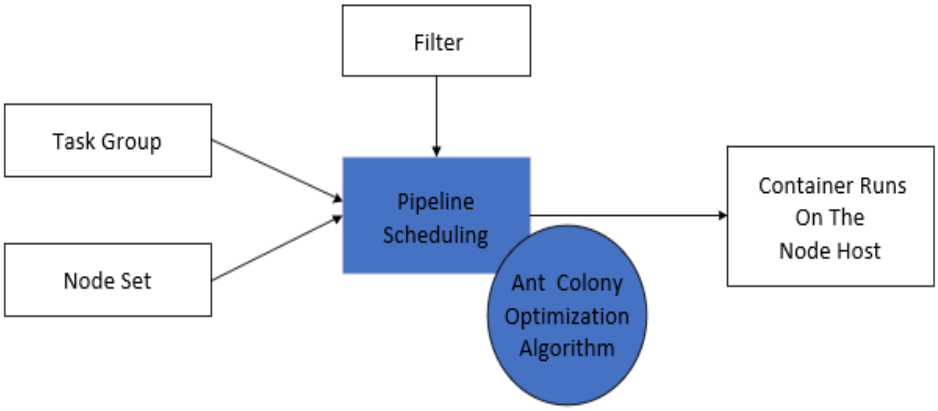

Figure 4.1 ACO scheduling structure diagram based on Swarm

4.2 Characteristics and improvement of ant colony algorithm

Ant colony algorithm is a heuristic intelligent algorithm, a probabilistic algorithm, that can be used to solve NP-complete problems. The algorithm is put forward by Italian researchers, they found that at the time of ant looking for food, has certain regularity, single ants foraging behavior is simpler, does not have the necessary of the study, and a group of ants foraging process is hidden wisdom, they can be in a different environment, to find a shortest path to food. The intelligent behavior depend on a change the parameters of the pheromone in ant, which is passed to adjust between ant colony of ants can perceive the pheromone concentration, the greater the concentration of ants will choose the path to crawl in search of food, and an ant walking the path is also left pheromones, pheromones can grow over time decreases continuously, thus implementing the feedback mechanism of ant picked his way, the ant colony can find a path to a food of the shortest path(Naik et al.,2016;Wu et al.,2001;Zhan et al.,2003;Zhu et al.,2004).

Ant colony algorithm has the following advantages compared with other swarm

intelligence algorithms:

1. This algorithm is applicable to a wide range of problems. If a problem can be represented by a connected graph and there is no high requirement for timeliness, this algorithm can be used to try it.

2. The randomness and certainty coexist in the solution space search of this algorithm.The heuristic function can be regarded as prior knowledge, and the value represents the possibility of choosing a certain path. In addition, the change of pheromone is a constraint on the heuristic function, and the change of pheromone can play an important role in whether the current path will be chosen, which reflects the certainty of positive feedback.Moreover, due to the existence of volatile factors, the previous path with not too low pheromone concentration can overtake the current excellent path, and expand the possibility of understanding space to search for excellent solutions, which reflects the randomness of negative feedback.

However, there are disadvantages to the original ant colony algorithm:

1. The operating mechanism of ant colony algorithm is based on roulette algorithm. In the initial stage of algorithm operation, lack of prior knowledge will make it take a long time to find an excellent solution.

2. When the search of solution space is iterated deeply, the exploration of solution space often comes to a standstill, or even a better solution cannot be found.

Since the two shortcomings are obvious, the above two shortcomings should be improved:

1. Appropriately increase the algorithm certainty in the whole algorithm cycle.The minimum task first completion algorithm is introduced to initialize the pheromone at the beginning, so that the algorithm can start from a task that needs the least time, and reduce the long time to search for the satisfactory solution.

2. The relationship between heuristic function and transition probability should be adjusted dynamically in each running stage of the algorithm, so that randomness and certainty can play a role together.Equilibrium factor (EF) is added to pheromone updates to guide decision making. In order to avoid the search falling into local optimum, the dynamic volatility coefficient is added. A global and local cooperation mechanism is adopted to dynamically balance pheromone updates. In this way, the local mechanism increases the flow speed of positive feedback information, while the global mechanism increases the possibility of spatial optimization.

4.3 System scheduling model design based on the improved ACO algorithm

The mathematical meaning of Swarm cluster scheduling is to create N container jobs C={C1, C2, C3..... CN} Selects the starting and running node locations.The mathematical definition is as follows:

$$t'_{ij} = \frac{I_i + O_i}{B_j} (4.1)$$

$$t''_{ij} = \frac{L_i}{M_j} (4.2)$$

$$t_{ij} = t'_{ij} + t''_{ij} (4.3)$$

In the above formula (4.1), is the input size of task I; is the output size of task ii; is the

time taken from task input to output result under a certain network bandwidth capacity.$I_i O_i t'_{ij}$In Equation (4.2), is the computation amount of the task, is the time taken for the container to start and run on the specified node.$L_i t''_{ij} B_j$The bandwidth capacity of host node j is the computing capacity of node j.M_jIn the formula, represents the total time taken for the container to create and run.t_{ij}

Minimal Task Finishes First Algorithm(Liu S et al.,2018) .The principle of which is the minimum amount of calculation work distribution to the best performance effect section point, at the beginning of the algorithm, first calculate the tasks provided resource nodes are expected to perform to complete takes time, namely each algorithm in the first task mapping relationship with the host node, and then rely on the good step matching task and the node to perform a task, after the completion of the task, the task set to remove it, will provide resources for each node queue is ready to receive the next task.The detailed execution process is as follows:

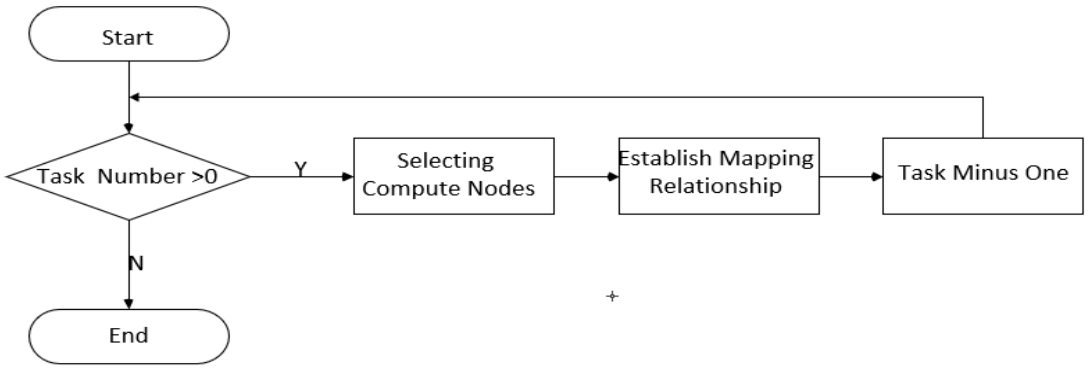

Fig. 4.2 Flow chart of execution

4.3.2 Deterministic and Random Function Initialization

In this study, the scheduling idea of the minimum task-to-finish algorithm is adopted. In the initial stage, the current task can be completed in a relatively short time, and has the characteristics of simplicity and reliability. Heuristic functions and pheromones(Liu S et al.,2018) .They are two important concepts in the whole life cycle of ant colony algorithm, and they are the embodiment of the advantages of randomness and certainty of the heuristic algorithm. Using this characteristic and introducing load balancing strategy, we can avoid the disadvantages of the algorithm.The heuristic function and pheromone are initialized using the scheduling results of the minimum task first completion algorithm.

$$F_{ij} = \frac{M_j}{M} + \frac{B_j}{B} \quad (4.4)$$

Equation (4.4) is the heuristic function update formula. Equation (4.4) represents the heuristic function used by ant colony algorithm scheduling.F_{ij}The overall performance of each resource-providing node is determined by the computing power

and bandwidth capacity of this point, where M represents the total computing power of the cluster and B represents the total bandwidth of the cluster.M_jRepresents the computing power of node j, and represents the bandwidth capacity of node j.B_j

$$F_{ij} = \frac{M_j}{M} + \frac{B_j}{B} + P \quad (4.5)$$

Formula (4.5) said the minimum first task schedule algorithm using the heuristic function, in which P said algorithm parameters using the algorithm of the strategy to the initialized heuristic function, reduce the blindness of ant colony, and assigned tasks to schedule made by the existence of the pheromone on the path for the initialization of was realized.

4.3.3 Update the implementation of iterative mechanism

In the actual production environment of the cloud platform, because the composition of cloud resources is relatively complex and heterogeneous, the hardware host is very different, and the container or virtual machine running on it will also run continuously, and the load is always changing.In these operating pressure, a container caused numerous task was assigned to a high-performance node, and the task is first to join the waiting queue, may be a long time not to perform, cause the quality of service, the user experience is inferior, but still have a lot of unused resources in cloud cluster nodes, these idle resources nodes due to limited resources and less to run a task, a container caused serious waste of resources.On the other hand, nodes with better performance are overloaded with tasks for a long time, resulting in reduced machine life of the container, and it is difficult to ensure that the execution time of a single task can reach the expected value.

$$EF = 1 - (C_x - C_{min})/(C_{max} - C_{min}) \quad (4.6)$$

In order to prevent the above problems, the balance factor () in formula (4.6) is used here to guide each iteration process of ant colony. Based on the last optimal solution, the load of each node in the last iteration is recorded in real time, and the last result is used as the basis for the pheromone update in the next iteration.EFThis takes the pressure off the overloaded nodes and assigns container tasks to the underloaded or idle nodes.

$$\tau_{ij}(t+1) = EF * \left((1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t)\right) \quad (4.7)$$

$$\tau_{ij}(t+1) = EF * \tau_{ij}(t) \quad (4.8)$$

In Equations (4.7) and (4.8), local and global pheromone updating mechanisms are adopted. If the ant selects a node in the current iteration task, the pheromone of this node will be updated through Equation (4.7); if no node is selected, the pheromone of this node will be updated through Equation (4.8).In equation (4.6), the parameter iteration cycle on the running time of the node X, said in the round of iterations is need least time node, said need time maximum node in the round of iterations, balance factor is introduced to guide the adjustment of the pheromone, found in the previous iteration cycle through the experiment, heavier load node will

be in the next iteration cycle pheromone concentration decline moderately.$C_x C_{min} C_{max}$Node with light load will also increase pheromone moderately in the next iteration cycle. In this way, the roulette mechanism in ant colony algorithm will be combined to achieve a more balanced load in the subsequent iteration process.

In the improved algorithm is used by the ant colony model proposed by Italian scholars Dorigo, parameters for volatile factors in the following formula, each iteration on the ant colony from start to finish I j pheromone increment, the initial phase of each path pheromone is zero, the formula (4.9) and (4.10) is used to update the iteration algorithm of global and local pheromone information$\rho\Delta\tau_{ij}(t)$ [24].(Liu W et al.,2016)

$$\tau_{ij}(t+1) = (1-\rho)\Delta\tau_{ij}(t) + \tau_{ij}(t) \quad (4.9)$$

$$\tau_{ij}(t+1) = (1-\rho)\Delta\tau_{ij}(t) + \tau_{ij}(t) + (4.10)\rho\tau_{ij}(t)$$

Formula (4.11) is used to calculate the pheromone increment, representing the total amount of pheromone secreted by k ants from starting point I to node j in t iteration tasks.$\sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t)$

$$\Delta_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) (4.11)$$

In order to explore the solution space, finding the optimal solution needs to speed up the pheromone flow to increase the probability of finding the optimal solution. The improvement of pheromone update can be divided into local, global and constraint.In an iteration cycle, regardless of whether the path chosen by the ant is good or bad, formula (4.9) should be used to update the pheromone on the path. Since the pheromone changes at the same time, the pheromone only accumulates and does not participate in volatilization.

When each iteration is completed, the node that takes the shortest task time in the current iteration cycle is used to strengthen the selected path with positive feedback. Formula (4.10) is used here to update the global pheromone.

When the pheromone increases to a certain level, it needs to be restricted. Due to the deepening of the level of iterative optimization, the pheromone may accumulate excessively, resulting in the decrease of the ability of ant colony optimization. Specifically, the pheromone will affect the priori function of the heuristic function, so it needs to be restricted by positive feedback. The use of local and global pheromones can speed up the flow and update of pheromones, reflecting the difference between some excellent paths and ordinary paths. The introduction of positive feedback constraints can increase the exploration of solution space and the possibility of finding high-quality solutions.

4.3.4 Algorithm flow

Below 4.3 is implementation steps of the algorithm, at the beginning of the algorithm, the need for enlightening function and pheromone initialization related operations, on the basis of experimental data to obtain inspiration function relation

with the weight of the pheromone, and the weighting factor are initialized, together with all the ants in the ant colony routing mechanism based on roulette wheel, then on to the host node selection secretion pheromones, to guide the next iteration, on a single ant to make a choice after the node will use formula (4.9) for local pheromone updating, the iteration cycle after all the ants on the path selection,Equation (4.10) is used to update the global pheromone, then the equilibrium factor for this iteration is calculated, which is then used to guide the next pheromone update, and the volatile factor is dynamically adjusted according to it until the end of the run.

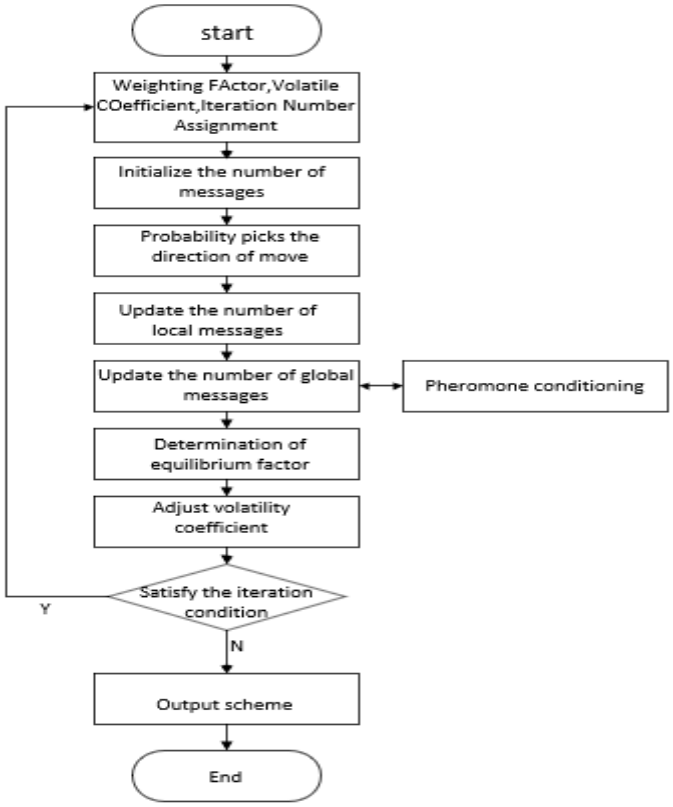

Fig. 4.3 Algorithm flow chart

The three parameters of ant colony algorithm are the key factors for it to play a good role. At present, it is not possible to obtain the optimal combination of the three parameters by general and applicable methods.α，β，ρ

1.ρWhen the weight factors are equal, the pheromone will play a small role in the path selection, so that the path sought by the ant last time will have little effect on the direction of the next time.$\rho\alpha$，βThis will lead to slow convergence of the algorithm. In the worst case, every ant wayfinding is almost independent of each other, and in this case, the algorithm cannot play its advantage.ρWhen the value is small, the increasing pheromone will cover up the role of the heuristic function, resulting in faster convergence of the algorithm, and the exploration of the solution space may be stalled.

2. The selection of is two parameters that have a constraint relationship with each other, which reflects the constraint and balance between the determinacy and randomness of the algorithm and requires a large number of experimental selection.α，β

3. The choice of R in the formula, R as molecules, the ants every iteration by pheromones produced much in the way of the important influence to the related operation data show that with the increase of R value, through the path pheromone increased significantly faster, which increases the uncertainty of the algorithm, can make the algorithm faster convergence, on the other hand, its have constraints.$\Delta\tau_{ij}^{k} = \frac{R}{L_k}\alpha,\ \beta,\ \rho$

4.4 Implementation Plan

After the optimized cluster scheduling strategy is introduced, the specific implementation plan will be introduced. Docker cluster is composed of two groups of nodes, namely management node and work node. Both are Docker hosts, and the management node is also responsible for task management, including scheduling decisions. Swarm is actually the real engine behind Docker's cluster management. Swarm puts the node information into a heap space and builds a minimum heap based on the number of containers currently on the node, which makes it extremely fast to find. The overall execution model of the improved ACO scheduling system based on Swarm is shown in Figure 4.4. Before scheduling, all work nodes will be processed through a set of filters. For example, the Constraint Filter only allows nodes that meet the specified constraints to be selected. Then, the improved ACO scheduling algorithm is used to replace the original algorithm to enter the pipeline scheduling task group. Specific by modifying two source files to join a group of optimization algorithm of modified code/docker swarmkit/manager/scheduler/indexed_node_heap. Go, add a function, define a constraint called multihostConstraint, mean different copies of the same service to fall on different host,Unlike other mandatory constraint, this is as far as possible to meet the constraint, modify search nodeHeap function, add a parameter serviceID, modify the code/docker swarmkit/manager/scheduler/scheduler scheduleTask function in the go.

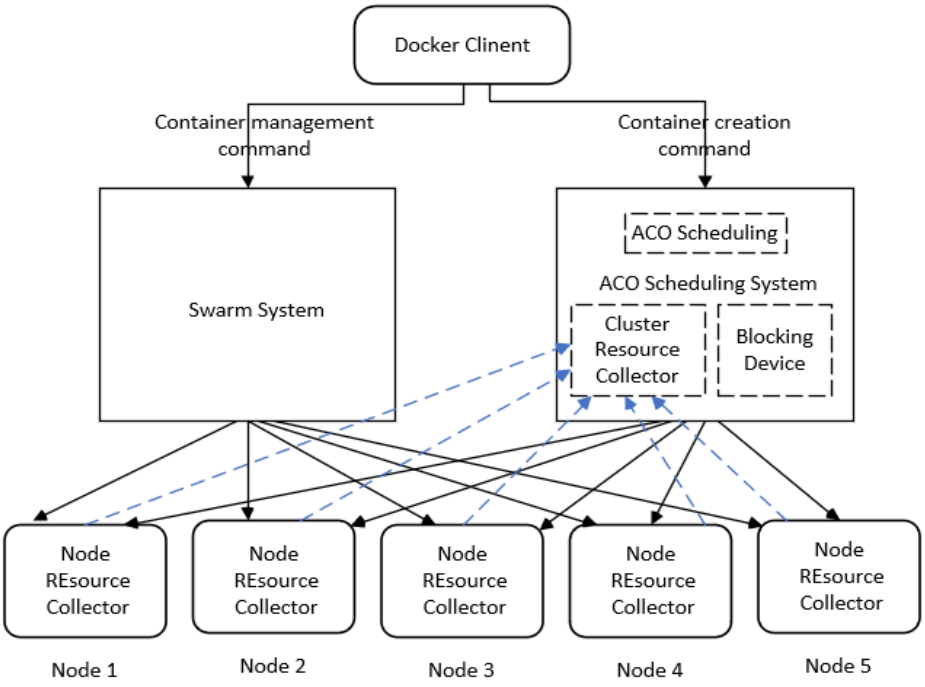

Figure 4.4 ACO scheduling strategy design diagram

4.5 Experiment and analysis

This experiment in real physical machine environment structures, cluster, the deployment of six Uubuntu14.04 version of the 64 - bit Linux system as a Docker host, Docker version for 18.09.9 - CE, using the Go language version 1.12, in order to reflect the distribution of scheduling resources, the resource configuration information given respectively according to table 4.1 deploy 5 host node Worker work, the rest one host Manager as the management node.

Table 4.1 Cluster node information table

Node number	System version	Number of CPU cores	memory
1	Uubuntu14.04	1	0.5 G
2	Uubuntu14.04	2	4G
3	Uubuntu14.04	1	1G
4	Uubuntu14.04	2	2G
5	Uubuntu14.04	1	0.5 G

Install Docker software on the cluster built by the above five machines to build Swarm cluster, and run the original Docker Swarm cluster scheduling and the improved ACO scheduling based on Swarm for a comparative test.

Select deployment of ordinary NGINX server as scheduling tasks, although at the time of modeling on the bandwidth capacity consideration, because the experiment scale smaller demand for network bandwidth (fixed), negligible, so this article only to the CPU and memory for the allocation of resources, NGINX server's CPU and memory reserved quantity were 0.5 and 128 MB. We had 12 NGINX servers up and running on five nodes as deployment containers. Deploying a unified task in this way makes it easy to observe the scheduling of the results.

By comparing the task scheduling of the two algorithms, it is easy to see that ACO achieves better load balancing than native scheduling.At the same time, the scheduling superiority of ACO can be reflected from the comparison of the NGINX pressure test data in Table 4.2.

Table 4.2 Comparison of NGINX pressure test data

The container no.	A native of Container load (requests per second)	Container load in ACO (requests/second)	The container no.	A native of Container load (requests per second)	Container load in ACO (requests/second)
Container 1	2163.26	3063.28	The container 7	1427.45	810.48
The container	1422.43	1736.53	Container 8	2281.35	1570.61

2					
Container 3	1208.46	870.38	Container 9	2244.87	2081.32
Container 4	1732.32	3065.77	Container 10	2236.5	2314.1
The container 5	2135.26	2791.28	Container 11	1285.56	3110.43
Container 6	2057.27	2718.93	Container 12	1942.56	2760.83

On the basis of the above experiment of two kinds of scheduling strategy of cluster to conduct stress tests, by using ApacheBench to conduct stress tests repeatedly measuring average data, ab performance test command "- 10 - n1000 c", the container number is not fixed, through cluster visualization tools portainer view, through comparing two groups of data on table 4.2, a native of each container load of 22137.29, the ACO combined in a container load of 26893.94, based on the original scheduling Spread,It can be seen that the improved algorithm improves the overall load of the original SPREAD cluster scheduling strategy by about 20%, and gives better play to the overall performance of the cluster.

5 Conclusion

An improved ant colony algorithm is used to optimize cluster resource allocation. Firstly, the original ant colony algorithm is improved, and the scheduling policy that the minimum task is completed first is used to initialize the pheromone to reduce the start time of search. Secondly by introducing balance factor, to coordinate the local and global pheromone pheromone update mechanism, the ant colony can according to the current iteration load and pheromone update mechanism to determine the next round of strategy, at the same time also introduced volatilization coefficient adjustment mechanism, to improve the global search ability of the algorithm, making optimal use of resources of each node. An improved ant colony algorithm was used to build the scheduling system model of SWARM. By adopting five Ubuntu16.04 version of the system to establish the virtual cluster network host, deployed Nginx server task as a container, then to pressure measurement of the whole cluster, the primary scheduling strategy and the improved ant colony algorithm has carried on the contrast experiment, through a comparison of the experimental results of multiple sets of analysis concluded that under the condition of current building cluster deployment task, it is concluded that to improve the ACO Swarm system pressure measuring the performance of original performance increases by about 20%.

Ethical approval

No need ethical approval.

Funding details

This work was supported in part by Guangdong Natural Science Youth Fund under Grant

（No.2020A1515110162), Key projects of social science and technology development in Dongguan under Grant (No. 2020507156156), in part by Special fund for science and technology innovation strategy of Guangdong Province under Grant (No. pdjh2020a1261).

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Informed Consent

No need informed consent.

Authorship contributions

Guanqaun Wu: Conceptualization, Methodology, Validation, Investigation, Writing.
Rongli Chen: Funding acquisition, Formal analysis, Visualization, Funding acquisition.
Desheng Zen: Methodology, Validation, Writing -Review & Editing, Supervision.

References

Cai Z(2017)Analysis of Container Isolation Based on Docker Technology .Electronic World, 2017(17):pp.195-195.

Docker Core Engineering, Docker 1.12(2016) Now with Built-in Orchestration! Jun.2016.Available:http://dockr.ly/2fejvo1

Li D（2019）Research on Scheduling Algorithm Based on Docker Cloud Platform .Xi 'an Polytechnic University.2019-05-08.

Li Z(2018) Design and Implementation of Elastic Load Balancing Function Based on Docker in Service Innovation Platform .Beijing postElectric university, 2018.

Ling J(2018)Research on Hadoop Small and Medium Cluster Performance Based on Docker .Nanjing University of Posts and Telecommunications,2018.

Liu G, Yu Y(2017)Analysis of Docker Container Technology .Secure Science and Technology, 2017(10):pp.28-32

Liu M, Gao C, Tian Y, Wang S, Liu L(2018)Scheduling Strategy Optimization Algorithm Based on Docker Swarm Cluster. Computer Department System applications, 2018, 27 (9) :pp. 199-204.

Liu S, LI Q, LI B（2015）Research on Container Isolation Based on Docker Technology .2015:pp.13-16.

Liu S, Zhang Q, Wang J(2018)An Improved Ant Colony Algorithm for Improving Cluster Scheduling Performance. Computer Systems shouldUse, 2018, 27 (7) : 173-176.

Liu W, Wang X, Qu H, Meng Y(2016)Research on Resource Scheduling of Server Cluster Based on Improved Ant Colony Algorithm. Micro ElectricityComputer Science and Technology,2016,33(03):98-101.

Lu T, Chen J, Shi J(2018)Research on Docker Security .Computer Technology and Development,2018,28(06):pp.115-120.

Lv Y(2020) Research and Implementation of Container Scheduling Strategy in Container Cloud Environment . Dalian University of Technology, 2020-06-02

Ma X, Liu Z（2015）A Scheduling Strategy and Algorithm Applicable to Docker Swarm Cluster .Computer Applications and Software.2015-05

Medel V,Tolosana-Calasanz R, Banares J,Arronategui U,Rana O(2018)Characterising resource management performance in Kubernetes. Computers andOf Electrical Engineering, 2018.

Naik N(2016)Building a virtual system of systems using Docker Swarm in multipleclouds.2016 IEEE International Symposium on Systems Engineering (ISSE).IEEE, 2016:pp. 1-3.

Nguyen N, Bein D(2017)Distributed MPI cluster with Docker Swarm mode, 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC),IEEE,2017:pp.1-7

Petrucci V , Laurenzano M , Doherty J （2015） Octopus-Man:QoS-Driven Task Management for Heterogeneous Multicores in Warehouse-Scale Computers. 21st IEEE Symp. on High Performance Computer Architecture. IEEE, 2015.

Wu B, Shi Z(2001)A segmented algorithm for solving TSP problem based on ant colony algorithm.Computer science Report, 2001 (12) :pp. 1328-1333.

Yang B, Dai W, Cao Y（2105）Docker Technology Introduction and Practical Practice .Machinery Industry Press,2015.

Yuan T（2019）Research on Task Scheduling Strategy in Container Cloud Platform . Guilin University of Technology, 2019-04

Zhan S , Xu J, Wu J(2003)Optimal selection of algorithm parameters in ant colony algorithm.tech-savvy Report, 2003 (5) : pp.381-386.

Zhang J, Xie T(2014)Research on Platform as a Service Architecture Based on Docker.Information technology and information technology. 2014 (10) : pp.131-134.

Zhu Q , Yang Z(2004)Ant colony optimization algorithm based on mutation and dynamic pheromone updating .Software to learn Report, 2004 (02) : pp.185-192.