



拟态通用运行环境的资源管理与调度技术

霍立田, 邵培南, 徐李定, 徐 骏

(中国电子科技集团公司第三十二研究所, 上海 201808)

摘 要: 为达到拟态通用运行环境(MCOE)对已/未知后门和漏洞主动防御、安全威胁攻击及时阻断和数据完整性有效保障等拟态防御目标,提出拟态资源调度准则,基于该准则从拟态资源管理与 MCOE 框架的交互设计、拟态资源管理与调度等方面论述拟态资源管理服务与调度算法的设计与实现,构造拟态运行节点软硬件资源异构特征分类器及基于三级异构度分类的节点 N 元组和 N 异构执行体元组,实现 N 异构执行体、服务器运行节点资源及其资源对象的随机性、动态性和异构性最大化与资源调度负载均衡,并通过拟态管理服务实例验证了云容器集群上拟态资源管理调度算法的正确性与有效性。

关键词: 拟态通用运行环境;拟态资源管理;资源状态;资源调度;N 异构执行体

开放科学(资源服务)标志码(OSID):



中文引用格式: 霍立田, 邵培南, 徐李定, 等. 拟态通用运行环境的资源管理与调度技术[J]. 计算机工程, 2020, 46(2): 159-169.

英文引用格式: HUO Litian, SHAO Peinan, XU Liding, et al. Resource management and scheduling technology for mimic common operating environment[J]. Computer Engineering, 2020, 46(2): 159-169.

Resource Management and Scheduling Technology for Mimic Common Operating Environment

HUO Litian, SHAO Peinan, XU Liding, XU Jun

(The 32nd Research Institute of China Electronics Technology Group Corporation, Shanghai 201808, China)

[Abstract] The main goals of mimic defense is to enable Mimic Common Operating Environment (MCOE) to implement active defense against known/unknown backdoors, block security threats and attacks in time, and ensure data integrity. To achieve these goals, this paper proposes the criteria of mimic resource scheduling. Based on the criteria, this paper analyzes the designing and implementation of mimic resource management services and scheduling algorithms in terms of the interaction design of mimic resource management and MCOE framework, mimic resource management, and mimic resource scheduling. This paper also constructs a heterogeneous feature classifier for software and hardware resources of mimic operating nodes, as well as a N-tuple and heterogeneous executor N-tuple based on the third-level heterogeneity. On this basis, this paper balances the resource scheduling loads, and maximizes the randomness, dynamicity and heterogeneity of N heterogeneous executors, resources on the running server node and resource objects. The correctness and effectiveness of the mimic resource management and scheduling algorithm on the cloud container cluster is verified using mimic management service instances.

[Key words] Mimic Common Operating Environment (MCOE); mimic resource management; resource state; resource scheduling; N heterogeneous executor

DOI: 10.19678/j.issn.1000-3428.0056123

0 概述

拟态信息系统是实现拟态化构造的信息系统, 基于邬江兴提出的拟态防御理论^[1], 为系统运行环境的基础层、应用支撑层和应用层(下文简称三层)

软硬件设施可能遭受的安全威胁攻击提供主动防御功能^[2], 确保处于威胁攻击状态下的系统功能正确可靠运行、运行过程数据的完整性和防窃取。拟态信息系统主要由系统客户端程序、应用服务程序功能等价的冗余异构执行体或包含冗余异构执行体

基金项目: 上海市科学技术委员会科研计划项目“拟态容器安全云平台研究”(18511104402)。

作者简介: 霍立田(1994—), 女, 硕士研究生, 主研方向为云计算、拟态防御; 邵培南, 研究员; 徐李定, 工程师; 徐 骏, 博士。

收稿日期: 2019-09-25 **修回日期:** 2019-11-05 **E-mail:** Lillian818@163.com

的云容器或虚拟机应用镜像(下文统称异构执行体)的集合、拟态防御架构^[3]和异构冗余集成化运行环境设施组成。

拟态信息系统按其开发语言可分为脚本和源语言两类,其中基于脚本的信息系统拟态防御产品已由信大开发^[4]并得到了广泛应用。拟态防御技术面向 JAVA 和 C/C++ 开发的 B/S、C/S 信息系统,基于上述组成进行开发和构建,其中拟态防御架构由拟态通用运行环境(Mimic Common Operating Environment, MCOE)实现,异构冗余集成化运行环境由支持 N 异构执行体、MCOE 各服务器运行的具有三层软硬件异构特征运行节点(如申威、飞腾、兆芯 CPU,红旗、麒麟操作系统, Tomcat、JBoss Web 容器)和构造化的拟态软件产品环境设施组成。

MCOE 提供应用服务请求的分发(N 异构执行体运行节点)、执行、表决和安全威胁诊断等服务功能,实现了服务请求执行过程的自动化,达到已/未知后门和漏洞主动防御、安全威胁攻击及时阻断、数据完整性确保等拟态防御目标。其由拟态分发器、N 异构执行体服务请求执行过程的内部和服务请求响应二表决器(下文称为内部和外部二表决器)、管理者和代理 3 类服务器组成。拟态分发器面向客户端应用服务请求,提供其接收、处理、分发以及 N 个异构执行体运行节点服务器的服务请求响应结果的接收、处理和转发客户端等功能。内部和外部二表决器分别针对服务请求执行过程中形成的 N 个关键结果数据(内部)和请求响应结果数据的表决调用,提供相应表决调用服务,实施表决结果的异常诊断和鲁棒性处理过程,反馈表决结果。管理者提供了拟态应用管理、拟态资源管理和拟态安全管理等功能。代理执行管理者命令,周期性或心跳地归集和

上报各运行环境节点的软硬件运行过程形成的数据(如日志和资源状态)。

本文主要从拟态资源的管理体系架构、状态管理、调度服务三方面进行论述。为达到调度的 N 异构执行体、服务器运行节点资源及其资源对象(云容器、虚拟机)的随机性、动态性和异构性最大化与资源调度负载均衡的目标,具体描述了拟态资源调度的算法设计与实现。

1 拟态资源管理与 MCOE 框架的交互设计

拟态资源管理提供拟态资源的状态管理和调度服务^[5],主要功能包括创建和实时维护集成化环境的运行节点资源、资源对象异构特征和状态全局映像(拟态资源状态管理);提供交互式资源和资源对象的状态和调度管理;基于实时的异构特征和状态全局映像,提供相应资源和资源对象异构特征最大化的 N 个异构执行体、分发器、内部和外部二表决器的运行节点资源和资源对象的调度请求服务。

资源调度^[6]服务请求响应结果主要包括 N 个异构执行体、运行节点、按需云容器或虚拟机。其中, N 个异构执行体的源代码以及相应运行节点三层软硬件的异构性,决定了拟态系统主动防御^[7]三层相应软硬件后门或漏洞安全威胁的水平^[4],异构执行体、运行节点资源和资源对象(如容器、虚拟机)的调度随机性和动态变化性能有效阻断黑客对系统的网络攻击,上述随机性、动态性和异构性最大化是拟态资源调度实现的目标。

因此,拟态资源管理(特别是资源调度)^[8]是 MCOE 拟态防御机制实现的基础和关键。作为拟态管理服务的组成,其与 N 异构执行体、MCOE 各服务器、集成化运行环境设施的关系见图 1,其中虚线框表示接口。

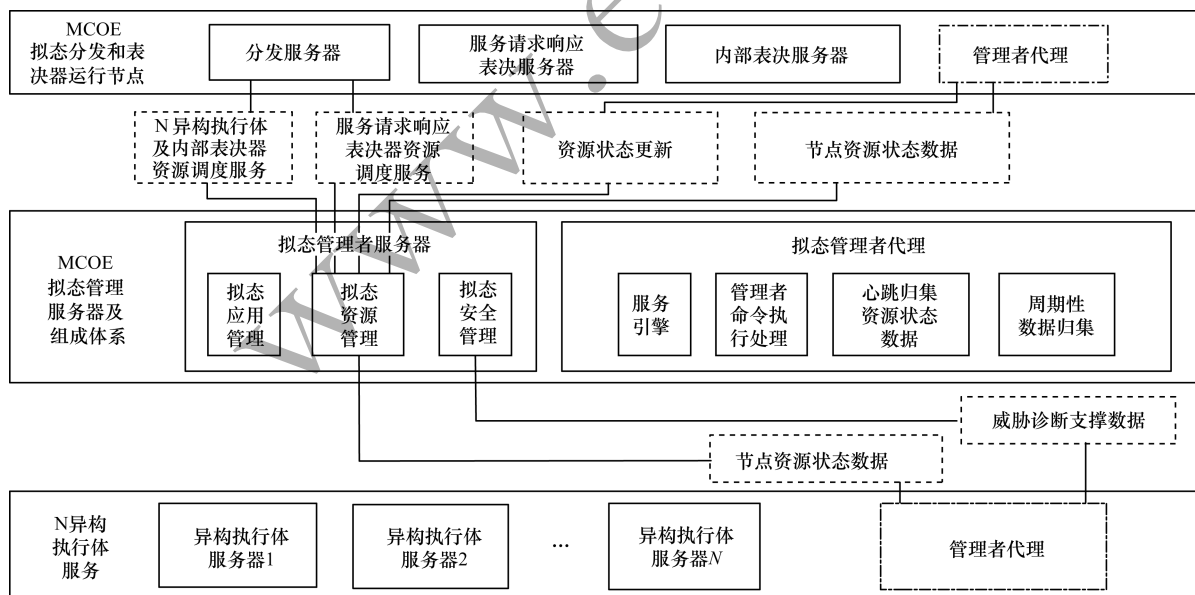


图 1 拟态资源管理与 MCOE 及其外部实体接口的交互关系

Fig. 1 Interactions between mimic resource management, MCOE, and its external entity interfaces

拟态资源管理基于 B/S 架构, 为应用客户端程序和 MCOE 其他服务器提供 Web 服务访问接口组件。拟态资源管理基于拟态管理 Web 服务器的管理者和管理者代理(M/A)架构, 实施基于命令策略

文件 profile 驱动的对象管理和资源状态的实时归集。管理者和管理者代理的接口遵循 MCOE 要求的 HTTP 通信协议, 其与外部主要接口的关系见表 1。

表 1 拟态资源管理与外部实体的主要接口定义

Table 1 Primary interface definition for mimic resource management and external entities

接口名称	主要数据元素和作用
N 异构执行体资源调度服务接口	分发器服务请求提出资源调度请求, 拟态管理(资源调度组件)返回 N 个异构执行体及运行节点的服务地址[IP, 云容器或虚拟机名称, 端口号]
分发器、内部和外部二表决器的资源调度服务接口	分发器按需提出分发器、内部和外部二表决器的资源调度请求, 拟态管理(资源调度组件)返回分发、内部和外部二表决服务器及运行节点的服务地址[IP, 云容器或虚拟机名称, 端口号]
心跳上报节点资源和资源对象状态数据接口	代理心跳实时上报节点资源和资源状态信息, 包括运行节点的 IP 和 CPU、各存储 I/O 等资源的使用量, 相应云容器或虚拟机的 CPU、各存储等资源的使用量; 管理者(资源状态管理组件)用于实时归集和更新运行节点资源异构特征和状态全局映像池

拟态资源调度现有研究工作主要基于 N 异构体随机性和相似度比较方法。文献[9]提出一种功能等价体调度装置及方法, 采取完全随机选择异构功能等价体为外部服务请求提供服务的方式, 降低了攻击者对某些漏洞和后门的探知或利用的可能性。文献[10-11]提出及改进的随机种子最小相似度算法(Random Seed and Minimum Similarity, RSMS)先随机选取冗余体, 再进行最小相似性比较。在实际拟态应用场景中, 异构执行体和运行环境软硬件的相似度可预先确定, 工程上需综合考虑 N 异构执行体和运行节点资源的随机性、动态性和异构性。

2 拟态资源管理服务框架设计

拟态资源状态管理为客户端提供资源状态查询、监管和调度功能, 框架设计如图 2 所示。

1) 节点资源归集和上报: 管理者代理心跳^[12]归集和上报分发、N 异构执行体、内部和外部二表决 4 类服务器节点以及相应云容器和虚拟机资源运行状态信息; 对于服务器或云集群, 基于集群服务器^[13-14] API 获取相应节点或云容器和虚拟机资源状态信息。

2) 映像和表维护: 拟态资源异构特征和状态管理基于上报的节点资源状态与节点资源状态的变化, 同步更新环境节点异构特征和状态的内存映像和表以及运行节点记录。

3) 资源调度: 拟态资源调度管理依据设计的调度算法为分发、N 异构执行体、内部和外部二表决服务器调度相应的运行节点或云容器、虚拟机计算资源。

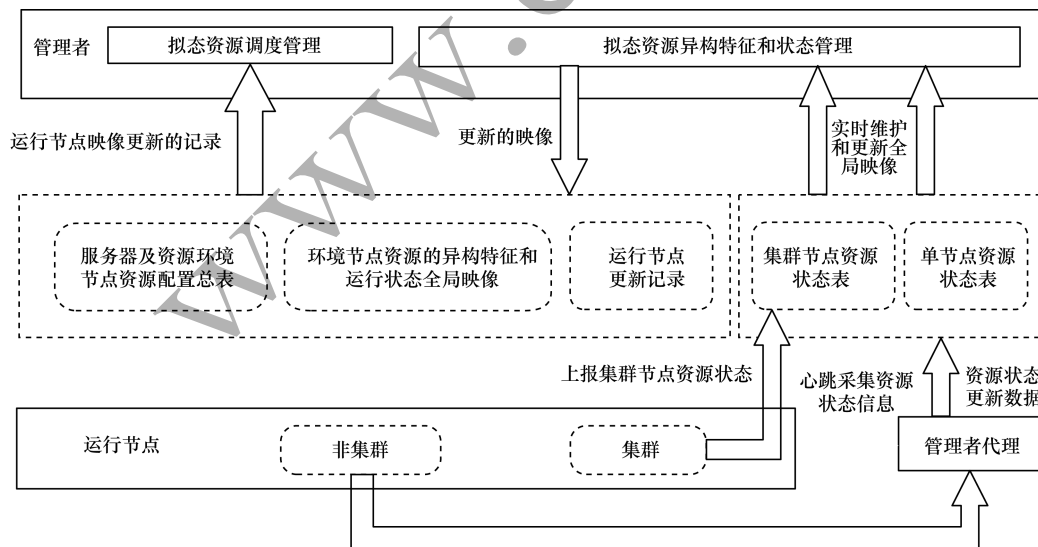


图 2 拟态资源管理框架

Fig. 2 Mimic resource management framework

3 拟态资源状态管理

拟态资源状态管理(如图3所示)具体功能如下:

1) 提供拟态系统运行环境节点资源状态整体映像和表的创建及实时维护,并提供交互式资源的调度和管理。

2) 基于拟态应用定义的运行环境节点异构特征表、节点及云资源分配表,各节点管理者代理心跳上报的上述各节点的资源使用状态,动态构建和维护

系统运行节点资源分配的总量及可分配量(全局资源状态内存映射,如CPU数量、CPU频率、内存容量等指标),并基于环境节点配置动态更新运行环境节点异构特征表和节点资源及其索引表,同时更新运行节点记录。

3) 管理者代理服务采用心跳方法上报获取的OS和云容器资源状态,由全局映像及表维护线程进行实时或按需的循环处理^[15]。表2为拟态系统运行环境下的节点资源状态映射。

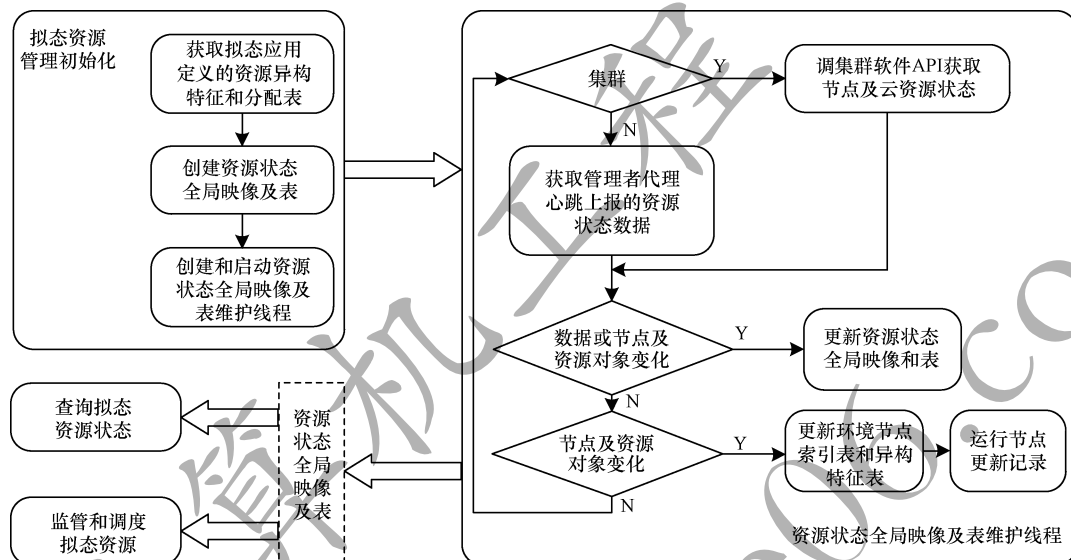


图3 拟态资源状态管理流程

Fig. 3 Procedure of mimic resource state management

表2 拟态系统运行环境下的节点资源状态映射

Table 2 State mapping of node resources in the operating environment of mimic system

运行节点 IP与域名	N异构执行 体部署方式	节点或容器 资源状态	CPU数量	CPU频率 /GHz	内存容量 /GB	磁盘I/O速率 /(MB·s ⁻¹)	CPU 占用率	内存 占用率	磁盘 I/O占用率
Y. X. M. N	服务器	节点资源状态	4	3.5	64	560	0.3	0.4	0.4
			2	2.5	16	210	0.6	0.7	0.5
			⋮	⋮	⋮	⋮	⋮	⋮	⋮
Y. X. M. K	云容器	容器资源状态	/	/	/	/	0.4	0.2	/
			/	/	/	/	0.5	0.4	/
			/	/	/	/	/
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

4 拟态资源调度服务

拟态资源调度服务主要为客户端服务请求的执行分别提供N异构执行体服务器、分发服务器、内部和外部二表决服务器的运行资源调度。其调度目标在于最大化地确保N个服务器运行节点三层软硬件的随机性、动态性和异构性,并实现资源的调度^[16]和负载均衡^[17]。

分发、内部和外部二表决服务器的资源调度是单一服务器运行资源的调度,是N异构执行体服务器的N=1特例,下文主要以N异构执行体为对象,描述资源调度的设计与实现。

4.1 拟态资源调度的基础数据结构

拟态应用为拟态资源管理创建了资源管理所需的基础数据结构,表3给出了三层软硬件通用异构特征的8位二进制数字化表示,其中,异构特征表征中1位为1、7位为0,最多表示8个异构特征,多于8个异构特征的情况,依相似度合并为8个表征。实际上大多数软硬件异构特征个数小于等于3。表4为分类索引构建的运行节点全局特征(其中“/”表示没有该值),基于此实施相应的记录分类,形成如表5所示的异构执行体分类索引。

表 3 拟态运行环境三层软硬件通用异构特征定义

Table 3 Common heterogeneous characteristics definition of software and hardware in three layers of mimic operation environment

层分类	软硬件	异构特征分类	异构特征最大值		异构特征	数字化值
			标志	数量		
基础层	计算机硬件	CPU 或不同厂家	CPUHMax	3	龙芯	00000001
					飞腾	00000010
					申威	00000100
	OS	Linux	OSHMax	2	红旗	00000001
应用支撑层	服务容器	Web 容器	WebHMax	2	麒麟	00000010
					Tomcat	00000001
					JBoss	00000010
	⋮	⋮	⋮	⋮	⋮	⋮
应用层	客户端 服务类程序	应用服务程序	SCodeHMAX	3	异构源代码 1	00000001
					异构源代码 2	00000010
					异构源代码 3	00000100
	编译	ComiplerParaHMAX	3	GCC + 参数 1	00000001	00000010
					GCC + 参数 2	00000010
					GCC + 参数 3	00000100

表 4 拟态系统异构环境下的异构特征映射

Table 4 Heterogeneous feature mapping in heterogeneous environment of mimic system

运行 节点 IP 与域名	运行节点 索引值	应用服务 程序部署 方式	异构特征分类索引标志							
			基础层 (I)			源代码 (J)		应用支撑层 (K)		
			层索 引值	PLHSAmount 个异构特征			SCodeHMAX 个异构源代码	层索 引值	ASLHSAmount 个异构特征	
				CPU	OS	⋯			Web 容器	⋯
Y. X. M. N	1	服务器	1	00000001	00000001	⋯	2	00000010	3	00000100
Y. X. M. K	2	云容器	2	00000100	00000010	⋯	3	00000100	1	00000001
Y. X. M. J	3	虚拟机	3	00000001	00000100	⋯	1	00000001	2	00000010
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

表 5 异构执行体分类索引

Table 5 Heterogeneous executor classification index

分类索引值	基础层索引值	源代码索引值	应用支撑层索引值	IP 地址/域名的 分类索引值	编译与参数索引值	编译与参数 异构特征值
1	3	2	4	3	2	1
2	2	1	3	1	1	3
⋮	⋮	⋮	⋮	⋮	⋮	⋮

4.2 拟态资源调度准则和资源部署规划需求

拟态管理资源调度服务基于如下调度准则实现 N 个异构执行体 (部署服务器、云容器或虚拟机) 和相应执行体运行节点资源及资源对象的调度算法:

准则 1 N 个异构执行体应符合源代码异构特征最大化原则, 具有调度的随机性。

准则 2 N 个运行节点基础层应符合软硬件异构特征最大化原则, 并确保异构执行体均衡分布在不同运行节点上。

准则 3 在遵循准则 1 和准则 2 的前提下, N 个运行节点应用支撑层软件应符合异构特征最大化原则。

准则 4 高优先级防御的软硬件异构特征最大

化应优先满足的原则 (由拟态系统应用管理给出)。

准则 5 资源调度动态变化性和负载均衡性。

N 异构执行体资源调度应遵循运行节点资源负载均衡, 并兼顾客户端按序服务请求的前后 N 异构执行体运行节点资源的动态变化性最大原则。

拟态资源管理遵循调度准则, 规划运行节点三层软硬件异构特征和负载的均衡配置及部署, 拟态资源调度算法基于该均衡配置和部署假设的前提下进行优化设计。

4.3 拟态资源调度概要设计

拟态资源调度服务基于构建特定系统应用服务程序的异构执行体 N 元组表及全局映像按需调度具

有异构性和负载均衡的 N 元组,包括运行节点、云容器和虚拟机。图 4 为拟态资源调度功能模块及交互示意图,反映了拟态资源调度(拟态资源调度初始化、拟态资源对象调度服务)两个部分的交互和功能。

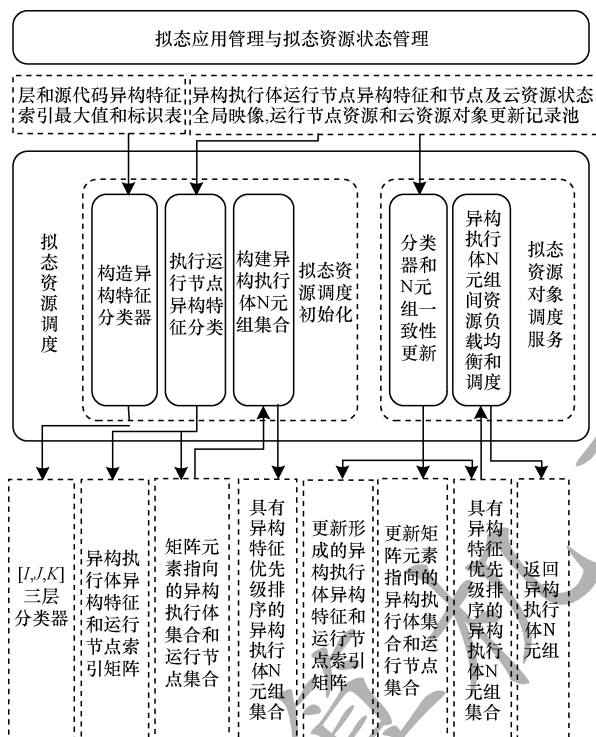


图 4 拟态资源调度功能模块及交互示意图

Fig. 4 Mimic resource scheduling function module and interaction diagram

4.4 拟态资源调度初始化

图 5 为拟态资源异构执行 N 元组的调度流程。

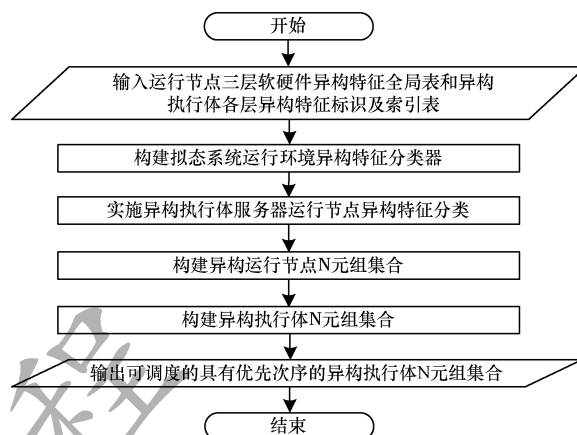


图 5 拟态资源异构执行体 N 元组调度流程

Fig. 5 Scheduling procedure of the heterogeneous executor N -tuple for mimic resources

4.4.1 异构特征分类器构建

构建的异构特征分类器^[19]见图 6,其中,层和源代码异构特征分类索引最大值和标志情况见表 6,三层软硬件异构特征最大值和标志情况见表 7,分类索引由 $[I]$ 、 $[I,J]$ 、 $[I,J,K]$ 、 $[I,J,K,L]$ 表示,每个索引指向运行节点集合和部署的异构执行体集合,分别表示经过上层分类结果形成的具有异构特征组合的分类标志索引,索引所指向的值包括运行节点索引集合和运行节点部署的异构执行体组合。表 8 ~ 表 10 表示相应 I 、 J 、 K 索引值所定义的异构特征。

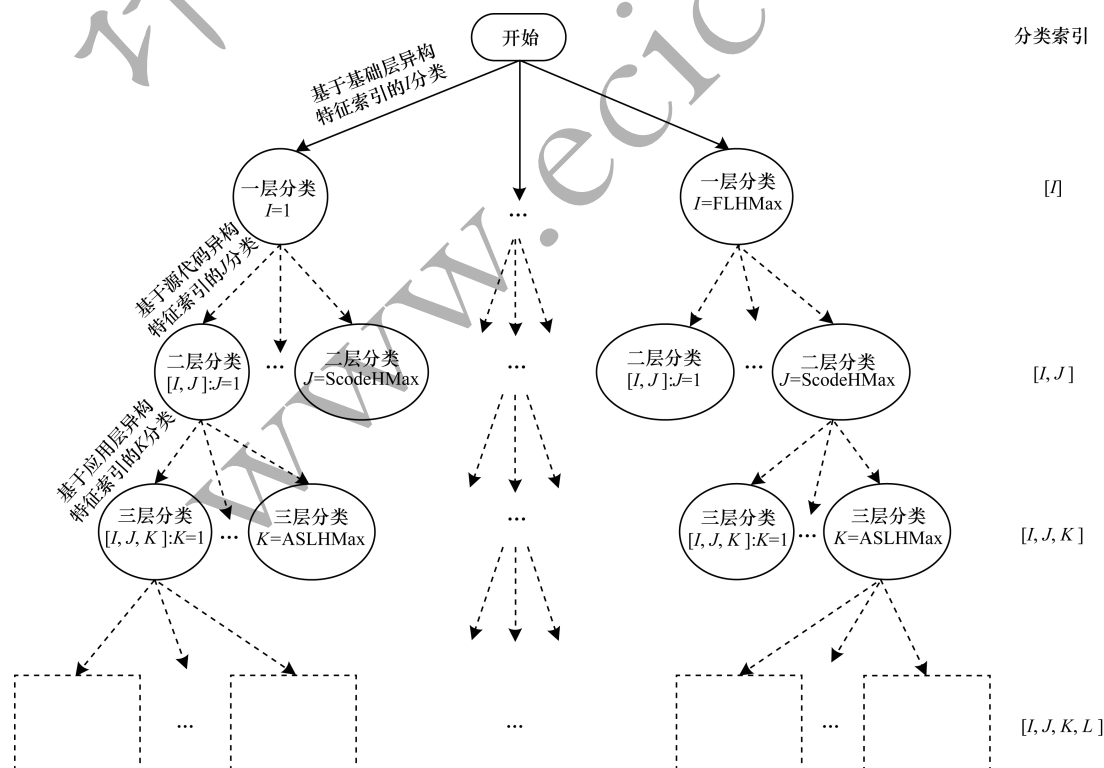


图 6 异构特征分类器分类示意图

Fig. 6 Classification diagram of heterogeneous feature classifier

表 6 层和源代码异构特征分类索引最大值和标志

Table 6 Maximum values and flags of heterogeneous feature classification indexes for layers and source code

层异构特征 索引分类	层软硬件数量 标志和数量值	异构特征 最大值标志	数值
运行节点	/	NodeMax	9
基础层异构特征索引	FLHSAmount	FLHMax	6
源代码异构特征索引	/	SCodeHMax	3
应用支撑层索引	ASLHSAmount	ASLHMax	2

表 7 三层软硬件异构特征最大值和标志

Table 7 Maximum values and flags of heterogeneous features of software and hardware resources of 3 layers

层分类	异构特征 分类	异构特征 最大数标志	数值
基础层	CPU	CPUHMax	3
	OS	OSHmax	2
	云容器	CloudContainerHMax	2
应用支撑层	Web 容器	WebContainerHMax	2
	⋮	⋮	⋮
应用层	编译 + 参数	CompileParaHMax	5
	异构执行体	ExecEntityHMax	3

表 8 基础层异构特征索引值和标志

Table 8 Index values and flags of heterogeneous features of the basic layer

分类索引值(I)	CPU	OS	云容器
1	00000001	00000001	00000001
2	00000010	00000010	00000010
⋮	⋮	⋮	⋮
FLHMax

表 9 源代码异构特征索引值和标志

Table 9 Index values and flags of heterogeneous features of the source code

分类索引值(J)	异构特征值
1	00000001
2	00000010
⋮	⋮
SCodeHMax	...

表 10 应用支撑层异构特征索引值和标志

Table 10 Index values and flags of heterogeneous features of the application supporting layer

分类索引值(K)	Web 容器	订阅和分发服务中间件	...
1	00000001	00000001	...
2	00000010	00000010	...
⋮	⋮	⋮	⋮
ASLHMax

4.4.2 运行节点异构特征分类及 N 元组构建

运行节点异构特征分类及 N 元组的构建算法具体如下:

输入 运行节点索引 $L(L=1,2,\dots,\text{NodeMax})$, 不同的索引代表不同的 IP 地址或域名, $\text{NodeG}[I](I=1,2,\dots,\text{FLHMax})$ 代表 I 分类后, 具有相同 I 索引值的运行节点索引值数组

输出 运行节点 N 元组集合

/* 面向服务请求的 N 异构执行体资源调度需求, 优先选择具有不同基础层异构特征的运行节点作为 N 元组元素, 构建 N 元组集合; N 大于基础层异构特征最大数 FLHMax 时, 先构建运行节点 M 元组集合 ($M = \text{FLHMax}$), 并基于各 $\text{Node}[I]$ 的运行节点数, 按比例均衡扩展集合中的各 M 元组到 N 元组; 最终均衡扩展 N 元组集合元素, 达到运行节点均衡分布, 以便于异构执行体 N 元组的构建 */

1. /* 判断 FLHMax 与 N 的大小关系 */

If $\text{FLHMax} > N$ then $M = N$ else $M = \text{FLHMax}$;

Repeat

从 $\text{NodeG}[I](I=1,2,\dots,\text{FLHMax})$ 中各 $\text{NodeG}[I]$ 选择不同 L 索引值组成的运行节点 M 元组;

until 遍历完成

2. 按比例均衡扩展

计算 1: $\text{NodeG}[I]$ 部署运行节点数比例 = $\text{NodeG}[I]$ 部署运行节点数 / 部署 NodeMax ;

For 所有运行节点 M 元组

{

$i = 0$;

$\text{NM} = N - M$

While $i < \text{NM}$ do

{

计算 2: M 元组集合中的 $\text{NodeG}[I]$ 运行节点数比例 = 属于 $\text{NodeG}[I]$ 中运行节点总数 / ($M * \text{集合个数}$)

$i++$;

选择 $\text{NodeG}[I]$ 运行节点计算 1 与计算 2 的比例偏差超阈值最大的 $\text{Node}[I]$, 优先选择 $\text{Node}[I]$ 指向的运行节点数组中不同的运行节点索引 L;

添加 L (运行节点索引值及标志) 成为 $M + i$ 个运行节点元素, 形成运行节点 $M + i$ 元组

}

/* 均衡扩展 N 元组集合元素, 达到运行节点比例偏差小于阈值 */

3. 计算 3: N 元组集合 $\text{NodeG}[I]$ 运行节点数比例 = N 元组集合中属于 $\text{NodeG}[I]$ 中运行节点总数 / ($N * \text{集合个数}$);

While 计算 1 与计算 3 的比例实际偏差大于阈值

{

针对误差大于阈值的 $\text{NodeG}[I]$ 集合, 按误差比例选择确定 $\text{NodeG}[I]$ 的运行节点数目, 优先选择不同 $[j,k]$ 的运行节点, 形成 N 元组

}

/* 针对运行节点服务器部署的多个异构执行体或云容器或虚拟机应用镜像执行体, 计算其应用支撑层和源代码异构度, 用于对运行节点 N 元组的排序和分层 */

4. /* 按 N 元组中的运行节点异构度对运行节点 N 元组集合排序和分层, 其中高优先级的节点 N 元组优先进行异构执行体 N 元组的构建 */

For 运行节点 N 元组集合中的各 N 元组

{

运行节点 N 元组异构度 = $\text{sum}(\text{该节点 N 元组中 N 个节点各软硬件异构特征异构度})$

(各异构特征的异构度 = 该节点 N 元组中节点上的所有异构执行体对应该异构特征的数字化值求 AND 结果中 1 的数量)

5. 合并相同运行节点 N 元组, 并按照 N 元组异构度

排序;

6. 根据异构度对节点 N 元组进行三层分层:可构成完全异构的 N 异构执行体的节点 N 元组为一层(各层异构特征数大于需求数的特征均不同,小于需求数的特征为该特征异构最大值),满足用户要求优先级次序的节点 N 元组为一层,可扩展构成的节点 N 元组为一层;

7. 根据记录的节点 N 元组的分层情况,优先调度可构成完全异构的 N 异构执行体的节点 N 元组进入下一步异构执行体 N 元组的构建模块;

}

8. 当增加节点或节点遭到破坏的情况下,直接在该步进行节点 N 元组的调整和更新分层(当有节点遭到攻击需要删除时,实时隐藏所有包含该节点的节点 N 元组;当有新节点添加时,根据新节点各层异构特征修改现有资源映射和表),以更好地满足基于现有系统资源和运行状态的动态调度

4.4.3 异构执行体 N 元组集合构建

异构执行体 N 元组集合构建算法具体如下:

输入 $[I, J, K, L]$ 组合索引; $ExecEntity[EE_{iJKL}]$ 为 L 运行节点、具有相同 I, J, K 异构特征值的异构执行体元素构成的数组, E_{iJKL} 为异构执行体元素个数,每个数组元素为 E , E 为异构执行体索引; $[I, J, K]$ 定义的各软硬件异构特征分类、顺序和最大异构特征数量 $XXXMax$

输出 异构执行体 N 元组集合

/* 遍历形成异构执行体 N 元组(按照运行节点 N 元组集合的各 N 元组优先级顺序,优先选择可构成完全异构的异构执行体层中的节点 N 元组) */

1. Repeat

对于节点 N 元组及其分层结果,优先考虑完全异构的节点 N 元组,从 Li ($i = 1, 2, \dots, NodeMax$, Li 为运行节点索引值)指向的 $ExecEntity[EE_{iJKL}]$ 集合中选择异构执行体,构建异构执行体 N 元组; N 元组每个元素包括 $[I, J, K, L, E]$ (E 为异构执行体索引)

/* 计算异构执行体 N 元组的软硬件异构特征最大化原则符合度 */

}

对 N 元组中的异构执行体,分别计算 I, J, K 各软硬件异构特征数量;对各软硬件异构特征八位二进制表示值进行 AND 操作,软硬件异构特征数量 = AND 结果的八位二进制中 1 的数量

软硬件异构特征最大化原则符合度 = 软硬件异构特征数量 / 软硬件异构特征 $XXXMax * 100\%$

/* 算法设定了缺省的软硬件异构特征优先级和权值,如指定优先顺序依次为源代码、CPU、OS、容器、Web 容器。其各指标最大值分别为 $SCodeHMax$ 、 $CPUHMax$ 、 $OSHMax$ 、 $CloudContainerHMax$ 、 $WebContainerHMax$,权值依次为 0.3、0.25、0.2、0.15、0.1,如用户定义了指定的特定一组软硬件异构特征优先级,其指定的软硬件按 70% 总权重和优先级分配,其他 30% 权重按缺省的优先级分配。*/

按软硬件异构特征优先级计算 I, J, K 的异构特征符合度 = SUM (各软硬件的权重 * $Min(1, \text{异构特征符合度})$)

}

Until 运行节点 N 元组遍历完成

2. 合并和去重异构执行体 N 元组,形成可调度的异构执行体 N 元组集合,每个 N 元组包括相应的 I, J, K 和软硬件异构特征符合度, N 元组中各异构执行体指向的 I, J, K, L, E (E

为异构执行体索引)

3. 按软硬件优先级和 I, J, K 异构特征符合度排序并分层

4.5 拟态资源对象调度服务

4.5.1 分类器和 N 元组一致性更新

基于分类树,当运行节点资源(增/减)和资源对象(异构执行体:容器/虚拟机增/减)动态变化时,将直接针对其各层特征进行分类器的插入和删除,对分类器和各层输出数据进行一致性更新,形成动态变化后的异构执行体 N 元组集合。

4.5.2 异构执行体 N 元组运行资源负载均衡与调度

异构执行体 N 元组运行资源负载均衡和调度算法的输入为运行节点和异构执行体 N 元组,输出为异构执行体 N 元组负载均衡队列和调度结果。图 7 为 N 元组运行资源负载均衡和调度活动图,当资源对象为容器时,需进行容器及其所在节点负载情况的综合考虑及计算。

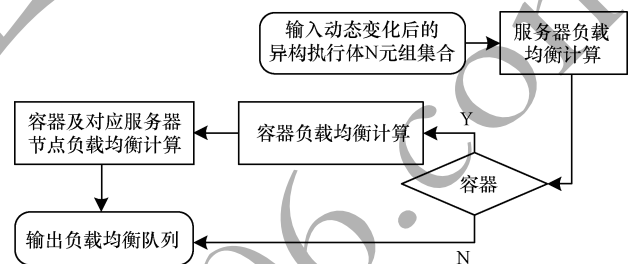


图7 N 元组运行资源负载均衡与调度流程

Fig. 7 Procedure of load balancing and scheduling of the N -tuple running node resources

4.5.3 异构执行体 N 元组间资源负载均衡与调度

异构执行体 N 元组间资源负载均衡和调度具体如下:

1) 服务器节点的负载均衡计算

图 8 为服务器节点负载均衡与调度流程。

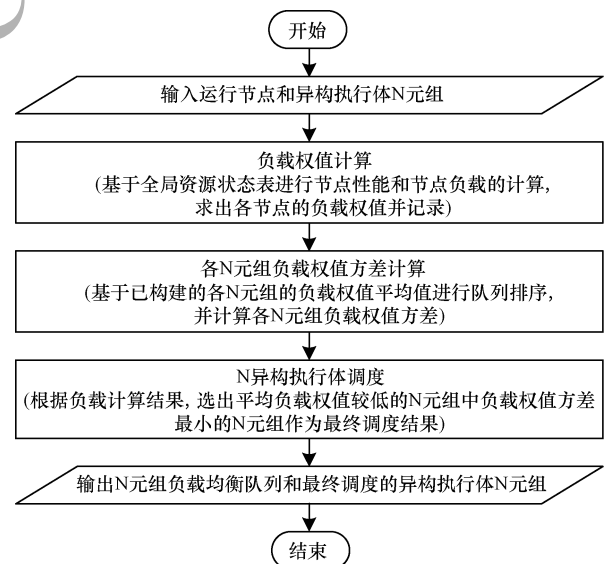


图8 服务器节点负载均衡与调度流程

Fig. 8 Procedure of load balancing and scheduling of the server node

对于 N 元组中的节点 N_i , 根据全局资源状态表可以获取其 CPU 数量 m 、CPU 频率 F_{C_i} 、内存容量 C_{M_i} 、磁盘 I/O 速率 R_{D_i} 等指标, 进而计算其性能 P_{N_i} :

$$P_{N_i} = k_1 m F_{C_i} + k_2 C_{M_i} + k_3 R_{D_i} \quad (1)$$

$i = 0, 1, \dots, n-1, \sum k_i = 1$

其中, k_i 是各项指标的权值参数, 反映不同类型的服务对各个指标的影响程度, 其和为 1, 该参数可以根据系统运行情况进行调节, 以期达到更好的效果。

同理, 其负载 L_{N_i} 主要从 CPU 占用率 O_{C_i} 、内存占用率 O_{M_i} 、磁盘 I/O 占用率 O_{D_i} 等指标进行考虑:

$$L_{N_i} = k_1 O_{C_i} + k_2 O_{M_i} + k_3 O_{D_i} \quad (2)$$

$i = 0, 1, \dots, n-1, \sum k_i = 1$

节点的负载权值 W_{N_i} 定义为节点负载 L_{N_i} 与节点性能 P_{N_i} 的比值, 采用式 (3) 进行计算。

$$W_{N_i} = \frac{L_{N_i}}{P_{N_i}}, i = 0, 1, \dots, n-1 \quad (3)$$

其中, W_{N_i} 越大, 说明节点负载越重, 其对应方差越小, 说明该 N 元组中的不同元素负载情况更均衡, 执行时异步概率更低。根据负载排序队列选取各 N 元组中负载权值及其方差最小的集合作为最终调度结果。

2) 云容器和虚拟机的负载均衡计算

云容器、虚拟机均需要基于物理服务器节点进行部署和分配, 其负载均衡调度方式类似, 本文以云容器为例进行设计。云容器的负载均衡与调度流程如图 9 所示。

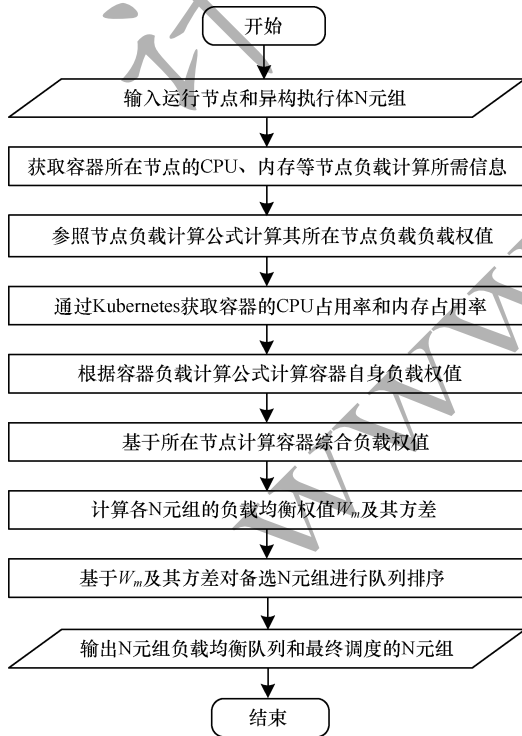


图 9 云容器负载均衡与调度流程

Fig. 9 Procedure of load balancing and scheduling of the cloud container

云容器负载均衡计算方法具体如下:

(1) 容器负载权值计算

参照物理节点的负载计算公式, 面向拟态信息系统访问程度, 根据节点资源对象状态映射获取容器 CPU 占用率 O'_{C_i} 和内存占用率 O'_{M_i} , 给出如下简化的容器负载权值 W'_{N_i} 计算公式:

$$W'_{N_i} = k_c O'_{C_i} + k_m O'_{M_i} \quad (4)$$

其中, k_c 、 k_m 分别为两项指标的权值参数, 反映不同类型的服务对各个指标的影响程度, 其和为 1。

对于基于云容器部署的异构执行体可调度性同时受其所在物理服务器节点的影响, 记节点负载权值所占比重为 k_n , 容器负载权值所占比重为 k_s , 其计算公式如下:

$$W_i = k_n W_{N_i} + k_s W'_{N_i} \quad (5)$$

(2) N 元组负载均衡计算

针对按序输入的各异构执行体 N 元组, 分别计算各个 N 元组中元素的负载 W_i , 取中位数 W_m 作为该 N 元组的负载性能, 多线程调度时的平衡度、响应度评价指标。

(3) 基于负载均衡的随机调度

选取输入的异构执行体 N 元组中同时满足 W_m 小于 0.7, 方差小于 0.5 的 N 元组。统计这些 N 元组当前的调度情况, 哈希选取调度次数小于中位数的 N 元组作为调度结果, 并对该 N 元组的调度次数加 1, 保证随机性。

4.6 实例分析

基于本文提出的拟态资源管理和 N 异构执行体调度方案进行简单的实例分析与验证。实例基础环境为包含 4 个节点、3 个基础支撑层 (包含 3 种 CPU、2 种 OS、2 种容器) 的异构特征, 异构执行体为 3 种异构源代码, 应用支撑层考虑现有两种 Web 容器 (Tomcat、JBoss); 对于编译异构特征, 假设已事先进行关联。

表 11 为实验所需的资源状态基础信息, 其数字化值可关联到本文资源管理部分的定义。目标为从其中选出 3 个可调度的满足异构性最大化的执行体。根据本文设计的调度算法, 通过分类器对现有资源状态进行分类, 首先得到 3 个不同的基础层异构执行体池, 其中 1 号池包含两个节点 (Node1、Node2), 根据节点 N 元组选择的方案, 本次实验阈值设为 0, 得到 4 个节点 N 元组, 分别是: 1) Node1, Node3, Node4; 2) Node2, Node3, Node4; 3) Node1, Node2, Node3; 4) Node1, Node2, Node4。经过异构度计算, 元组 1 和元组 2 属于可完全异构层, 其优先次序优于元组 3 和元组 4, 因此优先基于元组 1 和元组 2 的节点 N 元组构建异构执行体 N 元组, 并根据本文算法输出 12 个 N 元组集合及其异构度排序, 其中, 2 组异构度为 1 的异构执行体 N 元组 (完全异构), 1 组异构度为 0.95 的异构执行体 N 元组, 6 组异构度为 0.9 的异构执行体 N 元组……, 如表 12 所示为满足完全异构的 N 元组。

表 11 示例实验资源状态信息

Table 11 Sample experiment resource status information

基础层(I)				节点(L)	源代码(J)		应用支撑层(K)	
索引值	CPU	OS	容器		索引值	类型	索引值	Web 容器
1	龙芯	红旗	容器 1	1	1	源代码 1	1	Tomcat
				2	2	源代码 2	2	JBoss
					3	源代码 3	1	Tomcat
2	飞腾	麒麟	容器 2	3	2	源代码 2	1	Tomcat
					3	源代码 3	2	JBoss
3	申威	红旗	容器 2	4	1	源代码 1	2	JBoss
					2	源代码 2	1	Tomcat

表 12 满足完全异构的节点 N 元组特征

Table 12 Node N-tuple characteristics that satisfy complete heterogeneity

N 元组序号	异构度	负载度量		基础层(I)			节点(L)	源代码(J)	应用支撑层(K)	
		权值	方差	CPU	OS	容器			Web 容器	
1	1	0.3	0.26	龙芯	红旗	容器 1	Node1	源代码 1	Tomcat	
				飞腾	麒麟	容器 2	Node3	源代码 3	JBoss	
				申威	红旗	容器 2	Node4	源代码 2	Tomcat	
2	1	0.6	0.48	龙芯	红旗	容器 1	Node2	源代码 3	Tomcat	
				飞腾	麒麟	容器 2	Node3	源代码 2	Tomcat	
				申威	红旗	容器 2	Node4	源代码 1	JBoss	

对满足完全异构的两组异构执行体 N 元组进行负载均衡计算,经过实验建立的复杂场选定多个权值参数 $k_c = 0.5$ 、 $k_m = 0.5$ 、 $k_n = 0.4$ 、 $k_s = 0.6$,得到节点 N 元组 1 的负载权值为 0.3,方差为 0.26;节点 N 元组 2 的负载权值为 0.6,方差为 0.48;均小于设定的 W_m 和方差阈值,满足均衡性调度。当前状态两 N 元组调度次数均为 0,因此哈希^[20]选取其中任意一组作为调度结果,保证随机性、动态性和异构性三性最大化,增强攻击难度。

5 结束语

本文基于网络安全领域的拟态防御技术,针对 MCOE 运行环境中资源管理和调度问题,论述拟态通用运行环境的资源管理方案设计与实现,特别是针对现有资源状态的调度和实现。通过拟态管理服务的简单示例验证了对云容器集群上资源管理拟态调度方案的有效性。下一步将结合现有资源状态与 Kubernetes 调度机制实现拟态资源调度算法的优化和高效执行。

参考文献

- [1] WU Jiangxing. Introduction to cyberspace mimic defense[M]. Beijing: Science Press, 2017. (in Chinese)
邬江兴. 网络空间拟态防御导论[M]. 北京: 科学出版社, 2017.
- [2] SI Xueming, WANG Wei, ZENG Junjie, et al. A review of the basic theory of mimic defense[J]. Engineering Science, 2016, 18(6): 62-68. (in Chinese)
- [3] 斯雪明, 王伟, 曾俊杰, 等. 拟态防御基础理论研究综述[J]. 中国工程科学, 2016, 18(6): 62-68.
- [4] WU Jiangxing. Theory to cyberspace mimic defense[M]. Beijing: Science Press, 2018. (in Chinese)
邬江兴. 网络空间拟态防御原理[M]. 北京: 科学出版社, 2018.
- [5] TONG Qing, ZHANG Zheng, ZHANG Weihua, et al. Design and implementation of mimic defense Web server[J]. Journal of Software, 2017, 28(4): 883-897. (in Chinese)
全青, 张铮, 张为华, 等. 拟态防御 Web 服务器设计与实现[J]. 软件学报, 2017, 28(4): 883-897.
- [6] LI Wenliang. Key technology research of GPU cluster scheduling management system [D]. Wuhan: Huazhong University of Science and Technology, 2011. (in Chinese)
李文亮. GPU 集群调度管理系统关键技术的研究[D]. 武汉: 华中科技大学, 2011.
- [7] ZHAN Z H, LIU X F, GONG Y J, et al. Cloud computing resource scheduling and a survey of its evolutionary approaches[J]. ACM Computing Surveys, 2015, 47(4): 63.
- [8] WU Jiangxing. Cyberspace mimic security defense[J]. Secrecy Science and Technology, 2014(10): 4-9. (in Chinese)
邬江兴. 网络空间拟态安全防护[J]. 保密科学技术, 2014(10): 4-9.
- [9] SONI H B, SHAH A. 2nd International Conference on Signals, Systems and Automation (ICSSA 2011) & 1st International Conference on Intelligent Systems and Data Processing (ICISD 2011) [C]. [S. l.]: Universal-Publishers, 2011.
- [10] WU Jiangxing, LI Junfei, ZHANG Fan, et al. A device and method of heterogeneous function equivalent body scheduling: CN106161417B[P]. 2016-11-23. (in Chinese)

- 邬江兴,李军飞,张帆,等.一种异构功能等价体调度装置及其方法:CN106161417B[P].2016-11-23.
- [10] LIU Qinrang, LIN Senjie, GU Zeyu. Heterogeneous redundancies scheduling algorithm for mimic security defense[J]. Journal on Communications, 2018, 39(7): 188-198. (in Chinese)
- 刘勤让,林森杰,顾泽宇.面向拟态安全防御的异构功能等价体调度算法[J].通信学报,2018,39(7): 188-198.
- [11] ZHANG Jiexin, PANG Jianmin, ZHANG Zheng, et al. Executors scheduling algorithm for Web server with mimic structure[J]. Computer Engineering, 2019, 45(8): 14-21. (in Chinese)
- 张杰鑫,庞建民,张铮,等.面向拟态构造Web服务器的执行体调度算法[J].计算机工程,2019,45(8): 14-21.
- [12] ANDREWS G R. Concurrent programming: principles and practice[M]. San Francisco, USA: Benjamin/Cummings Publishing Company, 1991.
- [13] NIU Kan. Research on scheduling technology of cloud service resource based on game model[D]. Zhengzhou: PLA Information Engineering University, 2017. (in Chinese)
- 牛侃.基于博弈模型的云服务资源调度技术研究[D].郑州:解放军信息工程大学,2017.
- [14] XU Xin. Game theory based resource scheduling approaches of cloud computing[D]. Shanghai: East China University of Science and Technology, 2015. (in Chinese)
- 徐昕.基于博弈论的云计算资源调度方法研究[D].上海:华东理工大学,2015.
- [15] ZHANG Chengcheng. Research and implementation of container cluster management platform based on Docker[D]. Beijing: Beijing University of Posts and Telecommunications, 2019. (in Chinese)
- 张城城.基于Docker的容器集群管理平台的研究与实现[D].北京:北京邮电大学,2019.
- [16] LOGENTHIRAN T, SRINIVASAN D. Multi-agent system for energy resource scheduling of integrated microgrids in a distributed system[J]. Electric Power Systems Research, 2011, 81(1): 138-148.
- [17] CHEN Wenkai. Design and implementation of high concurrent Web system architecture based on Docker container[D]. Beijing: Beijing University of Posts and Telecommunications, 2019. (in Chinese)
- 陈文楷.基于Docker容器的高并发Web系统架构设计与实现[D].北京:北京邮电大学,2019.
- [18] ERGU D, KOU G, PENG Y, et al. The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment[J]. Journal of Supercomputing, 2013, 64(3): 835-848.
- [19] ZHOU Zhihua. Machine learning[M]. Beijing: Tsinghua University Press, 2016. (in Chinese)
- 周志华.机器学习[M].北京:清华大学出版社,2016.
- [20] KNUTH D E. The art of computer programming, section 6.2.2[M]. [S. l.]: TBS Press, 1997.

编辑 陆燕菲