第 46 卷 第 12 期 计 算 机 工 程 2020 年 12 月
Vol. 46 No. 12 Computer Engineering December 2020

·移动互联与通信技术· 文章编号：1000-3428(2020)12-0163-08 文献标志码：A 中图分类号：TP311

基于消息中间件的分布式网络扫描

胡栋梁[1]，秦晓军[2]，王晓锋[1]

（1. 江南大学 物联网工程学院，江苏 无锡 214122；2. 江南计算技术研究所，江苏 无锡 214083）

摘 要： 网络扫描是网络安全评测和网络管理的重要手段。针对传统单点主动扫描方法与 Zmap、Nmap 工具均存在带宽资源利用受限、扫描效率低与 CPU 使用率高等问题，结合消息中间件的分布式网络扫描技术，提出一种分布式网络扫描架构和任务调度算法。利用消息中间件实现信息同步和扫描结果回传，并构建一种分布式网络扫描任务调度模型。实验结果表明，相比传统的单点主动扫描技术，基于消息中间件的分布式网络扫描技术在保证扫描准确率的同时，CPU 使用率与扫描响应时间均降低约 10%。

关键词： 网络扫描；端口扫描；消息中间件；分布式调度算法；网络仿真

开放科学（资源服务）标志码（OSID）：

中文引用格式： 胡栋梁，秦晓军，王晓锋. 基于消息中间件的分布式网络扫描[J]. 计算机工程，2020，46(12)：163-170.

英文引用格式： HU Dongliang，QIN Xiaojun，WANG Xiaofeng. Distributed network scanning based on message middleware[J]. Computer Engineering，2020，46(12)：163-170.

Distributed Network Scanning Based on Message Middleware

HU Dongliang[1]，QIN Xiaojun[2]，WANG Xiaofeng[1]

（1. School of Internet of Things Engineering，Jiangnan University，Wuxi，Jiangsu 214122，China；
2. Jiangnan Institute of Computing Technology，Wuxi，Jiangsu 214083，China）

【Abstract】 Network scanning is an important means of network security evaluation and network management. The traditional single-point active scanning method and tools including Zmap and Nmap suffer from limited bandwidth resource utilization，low scanning efficiency and significant CPU usage. This paper proposes a distributed network scanning architecture and task-scheduling algorithm based on the distributed network scanning technology of message middleware. The algorithm uses message middleware to synchronize information and return scanning results，and constructs a task-scheduling model for distributed network scanning. The experimental results show that compared with the traditional single-point active scanning technology，the proposed distributed network scanning technology based on message middleware can ensure the scanning accuracy while reducing CPU usage and scanning response time by about 10%.

【Key words】 network scanning；port scanning；message middleware；distributed scheduling algorithm；network simulation

DOI： 10.19678/j.issn.1000-3428.0056018

0 概述

网络安全不仅是国家安全的重要组成部分，还关系到人民群众的财产和信息安全。网络扫描作为研究网络安全的第一步，其通过对网络主机进行扫描，使得运维人员了解到网络主机的安全配置和运行服务，从而发现并处理存在的安全威胁，同时对网络风险等级进行评估[1]。传统的网络扫描一般采用单点主动扫描，适用于层次简单、规模较小的局域网。由于应用性能需求导致网络规模不断扩大，也促使网络构成日益复杂。在实际应用中，运维人员通常将具有一定规模的计算机网络划分为若干个子网，通过设置 VPN 和防火墙实现私有网络的安全性。上述技术的应用虽然可提高网络的可用性，但是降低了网络的可管理性，同时还限制传统单点扫描技术对较大规模网络实施网络扫描的能力。

为进一步提高网络扫描的实时性和准确性[2]，本文提出一种分布式网络扫描架构和任务调度算

基金项目： 国家自然科学基金（61672264，61972182）；国家重点研发计划（2016YFB0800803）。

作者简介： 胡栋梁（1992—），男，硕士研究生，主研方向为网络仿真技术；秦晓君，高级工程师、博士；王晓锋，副教授、博士。

收稿日期： 2019-09-16 **修回日期：** 2019-11-26 **E-mail：** 18352538173@163.com

法。其中,分布式网络扫描架构采用三层架构,且主控节点可调度感知节点,而感知节点可跨区域部署,因此感知过程可以不受地域限制。任务调度算法则通过分析历史感知时间,找到最优感知节点进行调度。

1 相关工作

1.1 单点扫描

在单点扫描方面,经典性的扫描软件主要包括 Nmap[3] 与 Zmap[4],由于软件设计的局限性,两者都不具备分布式扫描的特点,无法进行大规模、跨内部网的网络扫描。文献[5]针对现有网络扫描技术中存在扫描静态性功能、分析评估功能等不足,在对现有扫描技术基本原理综合分析的基础上,对网络扫描技术智能化策略进行分析探讨,并构建一种符合智能化策略的网络扫描系统概念模型。文献[6]提出分块二分算法,合理设置 2 次提取 IPID 序列号间等待的基本延时,以提高 IPID 隐蔽网络扫描效率。文献[7]提出一种限制 RST 速率的空闲扫描方法,以逃避传统空闲扫描检测方法的检测。以上研究提出的扫描方案均是针对单点扫描方法以及逃避扫描检测机制进行改进,并未提出采用分布式扫描方案[8]来解决大规模网络扫描效率较低的问题。

1.2 分布式任务调度算法

在分布式任务调度算法方面,文献[9]结合网格计算环境构建一种描述网格体系结构的完全分布式模型,该模型着重于分布式负载平衡算法的建模和描述。利用模型检验对研究协议的不同性质进行形式化验证,并给出一组性能分析结果。文献[10]针对当前任务调度算法的资源利用率低、负载严重失衡等难题,提出基于负载均衡的任务调度优化算法,以提高分布式系统性能。该算法根据节点的性能实现任务的重新分配和调度,使得节点负载尽可能均衡合理。文献[11]提出一种基于纳什均衡联合调度策略的分布式强化学习算法,相比于静态调度算法,该算法利用更少的系统知识,即可使得调度器去主动学习任务到达和执行的相关先验知识,以适应相邻调度器的分配策略,促使调度器的策略趋向纳什均衡。文献[12]提出对异构分布式计算系统的形式化描述,并建立静态任务调度问题的理论体系。通过分析总结最长动态关键路径(Longest Dynamic Critical Path,LDCP)算法的核心思想及存在的不足,提出一种运用节点信息流量减少 CPU 空闲时间碎片的并行任务调度优化算法。

上述调度策略都是根据任务的权重或者负载进行任务分配和调度,并未考虑通信子网的通信时间,因此不能直接运用到网络扫描中。基于以上分析,本文将重点研究消息中间件[13]技术的分布式扫描框架,并寻找最优感知节点的分布式调度算法。

2 分布式网络扫描架构与通信流程

2.1 分布式网络扫描架构

本文采用一种三层模型[14]的分布式端口扫描架构,具体如图 1 所示。

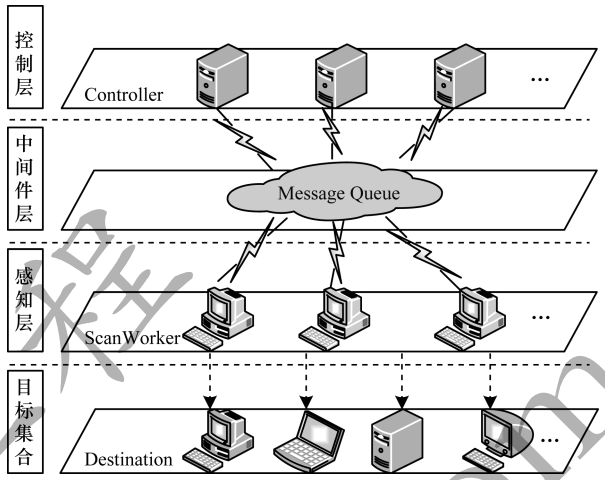

图 1 分布式网络扫描三层模型

Fig. 1 Three layer model of distributed network scanning

由图 1 可知,三层模型架构分别为控制层、中间件层与感知层,其主要特征是利用中间件层将控制层和感知层完全解耦,提升扫描系统的可扩展性与并发性。该架构的主要优势是将传统的单节点扫描扩展为多节点共同扫描,且多节点可处于跨区域不同的通信子网中。动态增加扫描节点数即可提升整个系统的扫描效率。此外,由于采用多节点机制,可利用分布式任务调度算法将给出的一系列扫描目标动态分发到距离最近的扫描节点,从而提高扫描目标节点的效率,降低扫描时间。目标集合是扫描的目标节点集合,其是大规模网络中运行 TCP/IP 协议的网络设备集合,主要包括主机、路由器、服务器与防火墙等网络设备,且该层不属于三层架构模型。三层架构的功能为:

1)控制层处于最上层,也是三层架构的核心,它控制着扫描任务的分配、定义、注册、调度与下发。相比传统的二层扫描架构[15],控制层并不直接将扫描任务发送给指定的感知层扫描节点,而是将所有的控制信息均经过中间件层间接地与感知层交互,其工作流程如图 2 所示,且具体步骤为:

(1)任务分配模块根据任务分配算法将用户定义的扫描任务分配到子任务中。

(2)任务定义模块将一个子任务以一个 task 对象来实现,task 是调度的基本单元。

(3)任务注册模块将会分配 task 一个全局唯一的 task ID,并将其持久化写入数据库。

(4)任务调度模块根据调度算法从数据库中取出子任务 task ID 并转交给任务下发模块。

(5)任务下发模块将消息队列中的任务消息经过消息中间件发布出去。

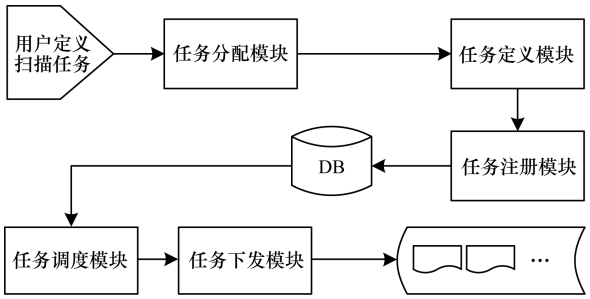

图 2　分布式网络扫描控制层工作流程

Fig. 2　Workflow of distributed network scanning control layer

2)中间件层[16]使用一种分布式应用间互相通信的消息队列技术,该技术可工作在磁盘或内存上[17],其中存储的任务消息可被控制层和感知层消费。通过使用消息中间件,可降低感知层扫描程序和控制层控制程序的耦合度,且感知层与控制层可相互独立工作。

图 3 中的消息中间件使用交换模块将任务下发模块下发的任务消息根据设定的 routing key 转发到不同的队列中,HostX 将消费其中的任务消息。队列的技术模型采用点对点 P2P 模型,具体如图 4 所示。该模型表示通过生产者 Client 调用 API 生产的消息 Message1 发送到 Queue 中,且只能被唯一消费者 Client 消费。因此,图 3 中队列 QueueX 消息只能被唯一指定的 HostX 消费,当一个任务消息经过交换模块进入 QueueX 中时,该任务消息将会排在队列末尾,当且仅当其前面的消息被扫描节点 Woker 执行后,该任务消息才会被执行。

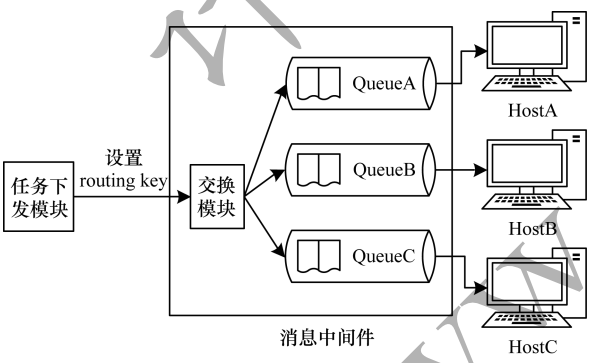

图 3　消息中间件流程

Fig. 3　Procedure of message middleware

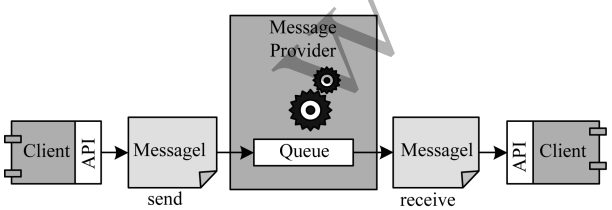

图 4　点对点模型

Fig. 4　Point to point model

3)感知层是三层架构中的最底层,其主要负责扫描任务的执行。感知层应用程序消费来自中间件层传递的控制消息,并根据消息获取扫描目标和扫描策略,并进行相应的扫描行为。分布式网络扫描感知层流程如图 5 所示,具体步骤为:

(1)接收任务消息模块获取中间层的任务消息,并且回传一个确认消息给中间层。

(2)分析模块根据 task ID 从注册管理器中获取扫描目标和扫描策略,且将分析信息发送至扫描器。

(3)根据扫描策略启动扫描器,扫描器的扫描技术分为使用 icmp 协议扫描、tcp 协议扫描以及 udp 协议扫描等,根据扫描策略使用扫描协议。扫描目标是目标主机的集合,即 IP 地址的集合,IP 地址有 2 种表示方法,具体如表 1 所示。

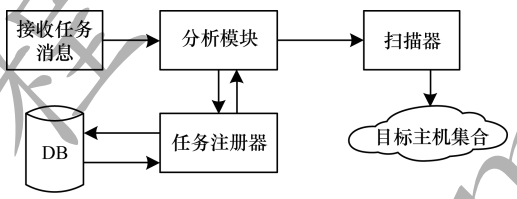

图 5　分布式网络扫描感知层工作流程

Fig. 5　Workflow of distributed network scanning awareness layer

表 1　IP 地址表示方法

Table 1　IP address representation

方法	表示形式
IP 集	[192.168.1.1,192.168.1.2,…,192.168.1.*N*]
子网表示	192.168.1.0/24

相比传统的控制层和感知层的两层架构,本文设计的三层架构具有以下特点:1)可跨平台、跨操作系统操作;2)具有良好的开放性和易扩充性;3)系统管理简单,具有较高的可用性;4)支持异种数据库和数据持久化。

2.2　分布式网络扫描通信流程

分布式扫描通信流程如图 6 所示,该流程主要包括描述各层包含的模块、各层之间信息流以及模块之间的信息流。它整体描述了扫描任务在该分布式系统中,从主控节点信息配置、任务定义注册、任务下发到任务执行的整套通信流程,具体步骤为:

步骤 1　主控节点信息配置,包括配置 broker_url 用于信息交换的消息队列 Message Queue,以及存储扫描结果的数据库 DataBase 与 result_serializer 设置传输数据格式。

步骤 2　主控节点对扫描任务定义与注册,扫描任务定义中使用任务分配算法将任务集划分成若干个互相独立的任务,并计算得到执行该任务耗时最少的扫描节点地址。然后将任务向注册管理器注册,注册管理器记录所有主控节点定义的任务 ID 和任务描述,并回复注册成功信息。

步骤 3　通过扫描任务的调度与下发,获取注册的任务 ID,主控节点利用任务 ID 和执行该任务的扫描节点地址构造任务消息,并调用下发管理器下发任务。

步骤 4　任务消息下发至消息中间件后,消息中间件根据任务消息中包含的扫描地址,通过路由功

能将任务消息转发至相应的扫描节点。

步骤 5 相应扫描节点获取从消息中间件中传递的任务消息,并从中解析获取任务 ID,扫描节点根据任务 ID 从注册管理器中获取具体的任务描述。

步骤 6 扫描节点通过获取到的任务描述,启动一个扫描线程执行该任务。当有多个任务到达时,则开启多个线程并发执行。

步骤 7 当扫描任务执行完成,获取扫描结果并将其存入扫描结果数据库,将任务完成信息回传至消息中间件并通知主控节点。

步骤 8 主控节点接收到任务完成信息后,从扫描结果数据库中提取结果信息。

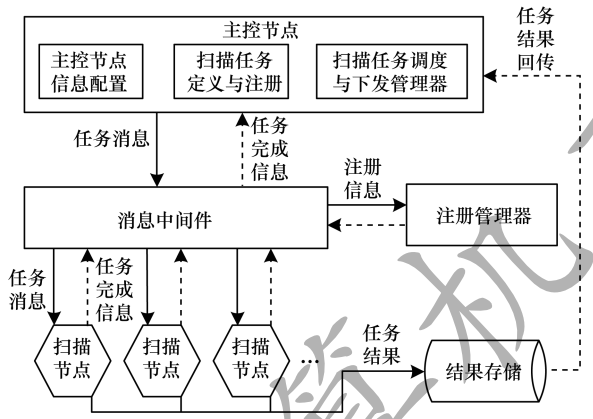

图 6 分布式扫描通信流程

Fig. 6 Communication procedure of distributed scanning

3 分布式扫描任务调度算法

基于上述分布式扫描框架,为进一步提高扫描效率,本文对面向扫描任务的分布式调度算法进行研究。

3.1 相关定义

定义 1 目标主机是网络扫描的目标。$A=\{a_1,a_2,\cdots,a_m\}$ 表示扫描目标地址集合,其中,m 为扫描目标的数量。

定义 2 扫描策略是用户指定的扫描类型的不同组合,$P=\{p_1,p_2,\cdots,p_n\}$ 表示需进行的扫描策略,其中,n 表示扫描类型的数量。扫描类型根据协议的类型不同可分为 icmp 协议扫描、tcp 协议扫描等,其扫描原理为:扫描节点 Client 使用不同的扫描类型向目标主机 Server 发送不同类型的报文,并根据返回的报文分析得出结果。扫描策略通过使用不同组合的扫描类型,分析获得目标主机的详细信息。

定义 3 扫描任务[18]主要包括目标主机和扫描策略 2 个部分。扫描任务 $H=(A,P)$ 是由扫描目标和扫描策略组成,在一个扫描任务中,不同的扫描节点对不同的目标主机进行网络扫描时,它们之间是彼此独立的,相互之间不需要消息通信,因此,不同目标的扫描可以调度到多个扫描节点中并行执行。本文将扫描目标作为任务划分的标准,并将一个扫描任务分解为多个独立子任务,且每个子任务就是对扫描目标进行指定扫描策略的网络扫描。因此,扫描任务集变换为 $H=\{h_1,h_1,\cdots,h_m\}$,其中,$h_i,i\in[1,m]$。

定义 4 扫描节点。单点主动扫描节点可根据用户指定的目标主机和扫描策略进行主机发现、端口扫描、服务发现和操作系统探测等网络扫描。在模型设计中假设扫描节点运行于硬件配置相同的机器中,因而可以假设各扫描节点的处理能力是相同的,扫描节点集合 $W=\{w_1,w_2,\cdots,w_r\}$,其中,r 为扫描节点的数量。

定义 5 扫描任务调度。采用任务调度算法将扫描任务调度到各个扫描节点中,该算法的目的是在最短时间内获取最准确的结果。

3.2 任务调度模型分析

图 2 控制层结构描述了调度模块的功能,下文通过图 7 的形式化语言详细描述调度模型。

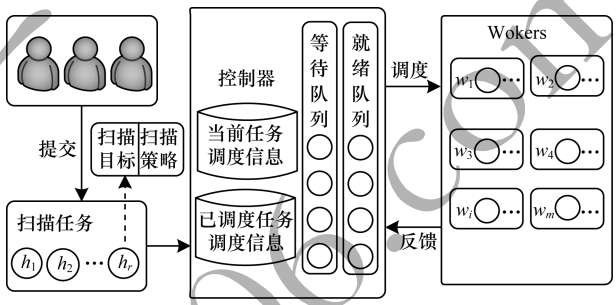

图 7 调度模型示意图

Fig. 7 Schematic diagram of scheduling model

与传统的单点扫描加入控制器模块相比,图 7 提出的调度模型将对扫描任务的控制管理与调度进行解耦,解决了由于扫描任务过多而造成程序阻塞、扫描效率降低的问题,且添加的就绪、等待队列使网络扫描从同步扫描转化成异步扫描,提升了程序调度子任务的性能。

扫描任务集合 H 和扫描节点集合 W 可以构造一个 $r\times m$ 的通信矩阵 $\boldsymbol{Z}$,用来表示扫描节点和扫描目标之间的通信时间。

$$\boldsymbol{Z}=\begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1m} \\ z_{21} & z_{22} & \cdots & z_{2m} \\ \vdots & \vdots & & \vdots \\ z_{r1} & z_{r2} & \cdots & z_{rm} \end{bmatrix}$$

其中,$z_{ij},i\in[1,r],j\in[1,m]$ 表示第 i 个扫描节点与第 j 个扫描目标之间的通信时间。需要特别说明的是:当扫描节点 i 与扫描目标 j 属于同一通信子网时,z_{ij} 为 0;当扫描节点 i 与扫描目标 j 不能建立连接或 j 不响应任何探测报文时,z_{ij} 为 ∞。

当扫描大规模网络时,进行一种扫描类型所需的时间为:

$$t(p_i)=u_i+c_it,i\in[1,n] \tag{1}$$

其中,u_i 表示执行某种扫描类型需要的 CPU 处理时间,c_i 表示探测数据包在扫描节点和目标主机间的

传输次数,t 表示探测数据包在目标主机和扫描节点之间的传输时间。扫描策略是多种扫描类型的组合,因此完整的执行时间为其包含的所有扫描类型的执行时间之和,表示方法如下:

$$t(P) = \sum_{i=1}^{n} (u_i + c_i t) \tag{2}$$

将通信矩阵 $\boldsymbol{Z}$ 代入式(2),可得:

$$t_{ij}(P) = \sum_{l=1}^{n} (u_l + c_l z_{ij}) = \sum_{l=1}^{n} (u_l) + z_{ij} \sum_{l=1}^{n} (c_l)$$
$$c_l > 0, i \in [1,r], j \in [1,m] \tag{3}$$

其中,$t_{ij}(P)$ 为扫描节点 i 对扫描目标 j 执行扫描策略 P 需要的时间。在同一扫描任务中,对其执行的扫描策略 P 是相同的,因此执行扫描策略 P 的 CPU 处理时间 $\sum_{l=1}^{n} (u_l)$ 相同,为简化公式用常数 α_1 替换,且 $\alpha_1 > 0$。由于扫描策略 P 是相同的,因此决定了探测数据包在目标主机和扫描节点之间的传输次数 $\sum_{l=1}^{n} (c_l)$ 相同,使用常数 α_2 替换,且 $\alpha_2 > 0$。由此,式(3)可简化为:

$$t_{ij}(P) = \alpha_1 + z_{ij}\alpha_2 > 0, i \in [1,r], j \in [1,m] \tag{4}$$

假设各个扫描节点 CPU 处理能力和传输次数是相同的,则分配给各节点的扫描子任务数量成为负载指标的主要考虑因子。本文使用 L 集合表示某时刻 t 各扫描节点的负载情况,$L(t) = \{l_1, l_2, \cdots, l_r\}$,$L_i(t)$,$i \in [1,r]$ 表示 t 时刻分配给第 i 个扫描节点的扫描子任务数量,$0 \leqslant l_i \leqslant m$。使用矩阵 $\boldsymbol{Q}_{m \times r}$ 表示扫描任务集调度方案,$\boldsymbol{Q}_{pi} = j$ 表示在第 i 个扫描节点上的第 p 位,将执行第 j 个子任务,其中 $i \in [1,r]$,p、$j \in [1,m]$。任务调度矩阵的列向量 $\boldsymbol{Q}_i$ 则为分配给第 i 个扫描节点的子任务集。

扫描任务完成时间取决于各个扫描节点执行完主控节点分配的子任务而花费的最长时间,因此可给定一个该扫描任务调度目标,即寻找扫描任务集 H 在扫描节点集 W 上执行时间最短的任务调度矩阵 $\boldsymbol{Q}_{\text{best}}$,使得花费时间最大的扫描节点的完成时间最小。

$$Y(\boldsymbol{Q}) = \min(\max Y_i), i \in [1,r] \tag{5}$$

其中,Y_i,$i \in [1,r]$ 表示第 i 个扫描节点执行完分配的子任务集需要的时间。

对于任意的扫描子任务 h_j,在理想情况下,应当将其分配到执行该扫描子任务最快的扫描节点上,而子任务 h_j 在扫描节点 w_i,$i \in [1,r]$ 上的完成时间包含 2 个部分:一部分是完成子任务 h_j 分配给 w_i 的任务集 $\boldsymbol{Q}_j$ 的执行时间,另一部分是 h_j 在 w_i 上的完成时间。

$$y(h_j, w_i) = \alpha_1 l_i + \sum_{1}^{l_i} z_{iq}\alpha_2 + \alpha_1 + z_{ij}\alpha_2 =$$
$$\alpha_1(l_i + 1) + \alpha_2 \left(\sum_{1}^{l_i} z_{iq} + z_{ij} \right), q \in \boldsymbol{Q}_i, l_i = |\boldsymbol{Q}_i| \tag{6}$$

由式(6)推出最快的完成时间为:

$$\min(y(h_j, w_i)) = \min(\alpha_1(l_i + 1) + \alpha_2 \left(z_{ij} + \sum_{1}^{l_i} z_{iq} \right) =$$
$$\alpha_1 \min(l_i + 1) + \alpha_2 \min \left(z_{ij} + \sum_{1}^{l_i} z_{iq} \right) \tag{7}$$

因此,式(7)即为将扫描子任务分配给扫描节点的判断标准。根据该标准,最优任务调度矩阵 $\boldsymbol{Q}_{m \times r}$ 需要满足以下约束条件:

1)主控节点必须将扫描目标分配至与其在同一通信子网的扫描节点中,即在通信矩阵 $\boldsymbol{Z}$ 中 $z_{ij} = 0$ 的扫描目标和扫描节点。

2)当存在多个扫描节点且满足上述条件时,式(7)中的 $\alpha_2 \min \left(z_{ij} + \sum_{1}^{l_i} z_{iq} \right)$ 是相同的,将其设为常量 A,则式(7)可转化为 $\alpha_1 \min(l_i + 1) + A$,根据扫描节点上的任务数量作为衡量扫描最快的标准,只需将子任务分配给任务数量最少的扫描节点,能够保证以最快的时间完成扫描任务。

3.3　基于历史扫描时间任务调度 Dscan 算法

根据上述分析,Dscan 算法将子任务分配给任务数量最少的扫描节点,能够保证以最快的时间完成扫描任务。相对于传统的单点扫描,该算法优化了扫描节点对目标主机的选择。

Dscan 算法描述如下:

步骤 1　for $i = 1$ to m do

在通信矩阵 $\boldsymbol{Z}$ 中,表明目标主机或其所在网络不可达的此类任务跳过,不对其执行,并将其从扫描任务集中删除。若存在唯一扫描节点与扫描目标在同一网络中,则直接将该扫描子任务分配给该扫描节点 w_i,并在任务调度矩阵 $\boldsymbol{Q}$ 中的第 j 列添加 h_i 的任务号。判断所有子任务是否已经分配完毕,若已分配完毕则转至步骤 4;否则未分配子任务为 m' 并转至步骤 2。

步骤 2　for $i = 1$ to m' do

1)寻找与所探测目标主机 h_i 处于相同子网的扫描节点 w_j,并将 w_j 加入到扫描节点子集合 w' 中。

2)如果 m' 为空,则跳出该循环体,否则继续执行下一步骤。

3)由于在 w' 中各个扫描节点与所探测目标主机 h_i 通信时间相同,则选择已分配任务数最少的扫描节点 w_j,$j \in [1,r]$,并将 h_i 分配给 w_j 执行,即在任务调度矩阵 $\boldsymbol{Q}$ 中的第 j 列添加 h_i 的任务号。

4)将相应扫描节点 w_j 的已分配数增加 1,并判断所有扫描子任务是否已经分配完毕,若已经分配完毕则转至步骤 4;否则未分配子任务为 m'' 并转至步骤 2。

步骤 3　for $i = 1$ to m'' do

1)计算扫描子任务 h_i 在各个扫描节点的完成时间 $y(w_j)$,$j \in [1,r]$。

2)找出能够使得 $\min(y(w_j))$,$j \in [1,r]$ 最小的扫描节点 w_{best},best $\in [1,r]$,将扫描子任务 h_i 分配到

w_{best}中,并将该子任务号添加至任务调度矩阵 $\boldsymbol{Q}$ 中。

步骤 4 确定任务调度矩阵 $\boldsymbol{Q}$,并完成子任务的分配。

4 实验验证与评估

4.1 实验环境

实验场景搭建在 Openstack[19] Queue 版本云计算管理平台上,各虚拟主机[20]使用的操作系统版本为 CentOS[21] Linux 7 Core Kernal 3.10.0-327.e17.x86_64,配置信息为 1v CPU、2 GB RAM 与10 GB DISK。虚拟 Router 使用的操作系统是搭载路由软件 Quagga[22] 的 Ubuntu[23] 12.04.5 LTS,配置信息为 1 V CPU、4 GB RAM 与 40 GB DISK。

目标网络拓扑模拟的是一个分布式网络环境,且 Dscan 系统部署在该分布式环境中。

4.2 扫描时间分析

在上述虚拟主机与路由器配置情况下,构建目标主机的发现场景,网络拓扑示意图如图 8 所示。其中,VM1,VM2,…,VM11 均为虚拟主机,R1 ~ R5 均为虚拟路由器,“云”表示主机所在网络。通过 Openstack 搭建上述网络测试环境,3 种扫描方法部署如表 2 所示。

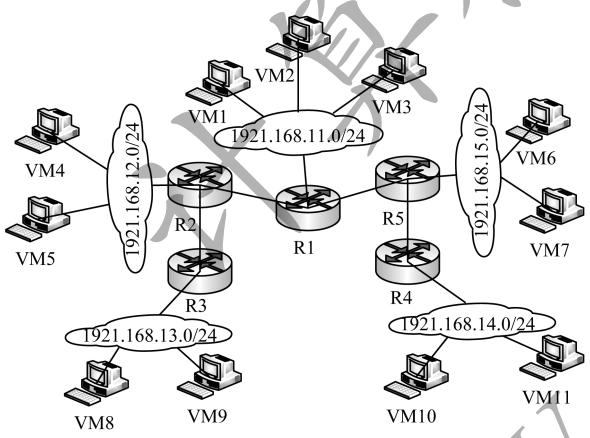

图 8 网络拓扑示意图

Fig. 8 Schematic diagram of network topology

表 2 3 种扫描方法中节点部署描述

Table 2 Description of node deployment of three scanning methods

扫描方法	主控节点	扫描节点
Dscan	VM1	VM4,VM6,VM8,VM10
Nmap	无	VM1
Zmap	无	VM1

3 种扫描方法针对的目标网络分别是 192.168.11.0/24、192.168.0.0/20、192.168.0.0/16、192.160.0.0/12 与 192.0.0.0/8,在对存活主机和非存活主机进行主机扫描时,扫描协议均采用 ICMP 协议,记录 3 种扫描方法对目标网络的扫描时间,结果如图 9 所示。

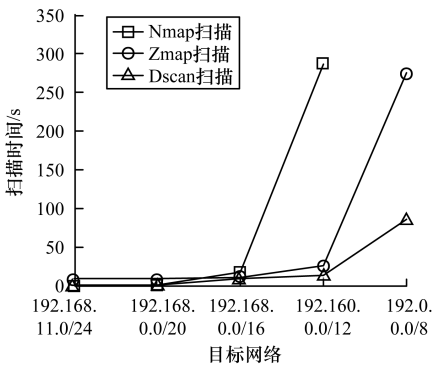

图 9 3 种扫描方法的扫描时间对比

Fig. 9 Comparison of scanning time of three scanning methods

从图 9 可以看出:当扫描主机个数小于 4 096 个(即目标网络为 192.168.11.0/24、192.168.0.0/20)时,Dscan、Nmap 的扫描时间几乎是一致的,且均低于 Zmap;当扫描主机个数大于 65 535 时,Dscan 的扫描时间保持在 1 min 以下,而 Nmap 和 Zmap 的扫描时间有不同幅度增长;当扫描网络为 192.0.0.0/12 时,Dscan 可在 2 min 内完成扫描,然而 Nmap 和 Zmap 无法在该时间内完成扫描。通过该实验验证了分布式扫描系统 Dscan 相比传统的单点扫描可提高扫描效率,同时也验证了本文提出的分布式框架中任务分配方案的可行性。

4.3 CPU 占用率分析

根据网络拓扑部署实验环境,分别利用 Dscan、Nmap 与 Zmap 对目标网络 192.0.0.0/12 进行扫描,运行过程均在主机 VM1 上进行,且以运行10 s、20 s、30 s、40 s、50 s 与 60 s 为采集点,所得扫描程序的 CPU 使用率如图 10 所示。从图 10 可以看出:在运行时间大于 10 s 后,Dscan 扫描的主控节点在 VM1 上的 CPU 使用率从 16% 逐渐降至 1%,这是因为 Dscan 主控节点在 10 s 前需要计算将各个扫描任务分发至其他扫描节点的时间,当分发任务完成后,主控节点异步等待其他扫描节点完成扫描任务,主控节点只需维持与各个扫描节点的连接即可,而其他 2 种扫描方法的 CPU 使用率在运行期间一直保持为 16% 和 58%,远高于 Dscan 的 CPU 使用率。通过该实验验证了分布式扫描方法 Dscan 相比传统的扫描方法可降低主控节点 CPU 资源使用率,达到节省资源的目的。

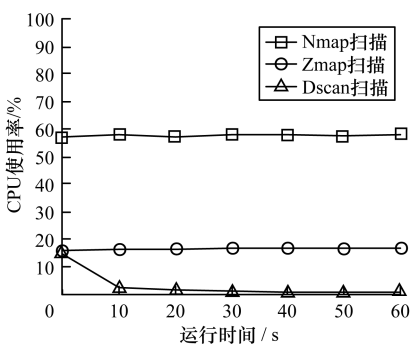

图 10 3 种扫描方法的 CPU 使用率对比

Fig. 10 Comparison of CPU usage of three scanning methods

4.4　分布式调度算法的有效性

根据互联网 IP 地址分配中心取得网络号,实验将其划分为 8 个网络区域,且主控节点和扫描节点分别为 116.193.16.5 与 xxx.xxx.xxx.3。图 11 为在该网络中存在的 8 个网络区域,且每个区域存在若干主机。本文将扫描每个区域的网络设备情况设置为一个扫描任务,并将其下发给主控节点与扫描节点。

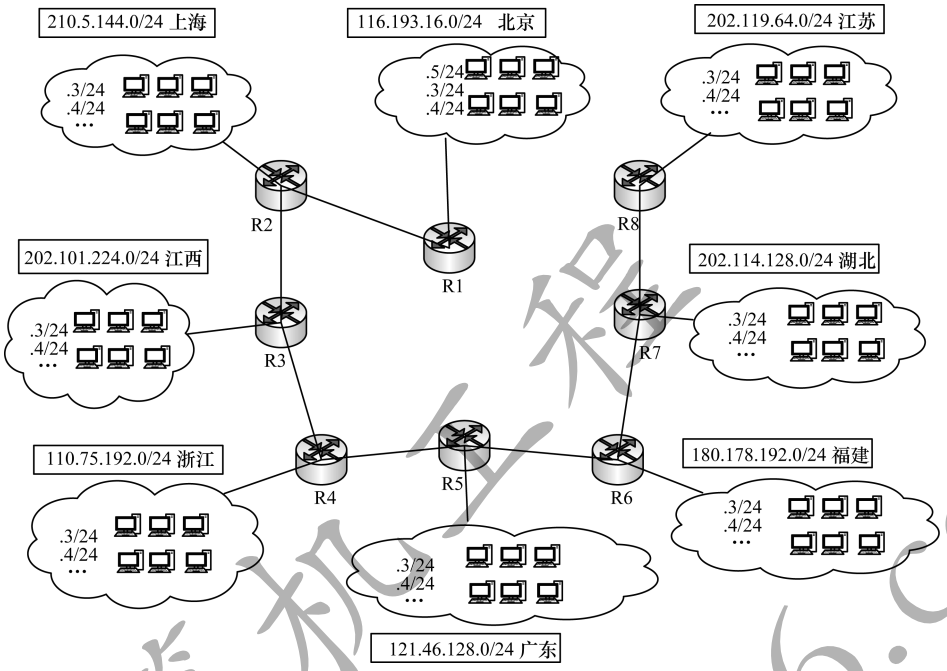

图 11　网络区域示意图

Fig. 11　Schematic diagram of network area

在不使用 3.3 节所述的分配算法时,扫描任务 h_j 随机下发给节点 w_i,$i \in [1,r]$。而经过上述分配算法后,扫描任务将会下发给与扫描任务相同的通信子网或者通信时间最少的通信子网。本文算法使用前后的响应时间对比如图 12 所示。从图 12 可以看出:当扫描区域数量为 1、2 时,本文算法使用前后的响应时间均为 50 s,上下波动不到 1 s;随着扫描区域数量的增多,响应时间相差逐渐增大,且当扫描区域数量增大至 8 时,采用本文算法后响应时间增加至 100 s,远低于未使用调度算法的 290 s。该实验可表明通过使用分布式扫描任务调度算法可提高分布式扫描效率,由此验证了本文所提调度算法的有效性。

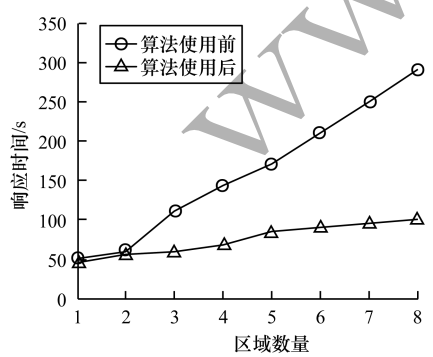

图 12　本文算法使用前后的响应时间对比

Fig. 12　Comparison of response time before and after the algorithm used in this paper

5　结束语

本文基于分布式网络扫描思想,对中间件异步通信技术进行研究,构建一种由控制层、中间层与感知层组成的三层分布式网络扫描框架,提出一种扫描任务调度算法。通过多组实验验证了分布式网络扫描技术相比传统单点扫描技术可有效提高扫描效率,降低单台主控端的 CPU 使用率。后续将对端口扫描技术进行进一步研究,在保证分布式网络扫描有效性的情况下,提高扫描效率与准确性。

参考文献

[1] LIU Jian, SU Purui, YANG Min, et al. Software and cyber security—a survey[J]. Journal of Software, 2018, 29(1):42-68. (in Chinese)
刘剑,苏璞睿,杨珉,等. 软件与网络安全研究综述[J]. 软件学报,2018,29(1):42-68.

[2] YANG Zhongyi. Research and implementation of key technologies of network security scanning system[D]. Changsha: National University of Defense Technology, 2007. (in Chinese)
杨忠仪. 网络安全扫描系统关键技术的研究与实现[D]. 长沙:国防科学技术大学,2007.

[3] LYON G F. Nmap network scanning: the official Nmap project guide to network discovery and security scanning [EB/OL]. [2019-08-10]. http://www.doc88.com/t-99193.html.

[4] DURUMERIC Z, WUSTROW E, HALDERMAN J A. Zmap: fast Internet-wide scanning and its security applications [EB/OL]. [2019-08-10]. https://www.usenix.org/sites/default/files/conference/protected-files/durumeric_sec13_slides.pdf.

[5] ZHAO Hanyun, LU Songnian, QI Kaiyue. Research on intelligentization of network scan technology [J]. Computer Applications and Software, 2008, 25(3): 184-185. (in Chinese)
赵汉云,陆松年,齐开悦. 网络扫描技术的智能化研究[J]. 计算机应用与软件,2008,25(3):184-185.

[6] ZHAO Qiu, HU Huaping, YU Haiyan. Theory and implementation of IPID covert network scanning [J]. Journal of Computer Applications, 2005, 25(4): 870-873. (in Chinese)
赵秋,胡华平,余海燕. IPID 隐蔽网络扫描的原理与实现[J]. 计算机应用,2005,25(4):870-873.

[7] ODHAVIYA R, MODI A, SHETH R, et al. Feasibility of idle port scanning using RST rate-limit[C]//Proceedings of the 10th International Conference on Security of Information and Networks. New York, USA: ACM Press, 2017: 224-228.

[8] GEORGE C, JEAN D, TIM K, et al. Distributed systems: concepts and design[EB/OL]. [2019-08-10]. https://cs.fit.edu/~pkc/classes/dc/slides/ch3.pdf.

[9] HAFAIEDH I B. A generic formal model for the comparison and analysis of distributed job-scheduling algorithms in grid environment[J]. Journal of Parallel and Distributed Computing, 2019, 132: 331-343.

[10] FAN Cheng, SU Ruofan. Task scheduling optimization algorithm based on load balancing[J]. Computer Engineering and Design, 2017, 38(6): 1532-1535. (in Chinese)
樊程,苏若凡. 基于负载均衡的任务调度优化算法[J]. 计算机工程与设计,2017,38(6):1532-1535.

[11] TONG Zhao, XIAO Zheng, LI Kenli, et al. Scheduling algorithm in distributed systems based on non-cooperative game[J]. Journal of Hunan University (Natural Sciences), 2016, 43(10): 139-147. (in Chinese)
童钊,肖正,李肯立,等. 分布式系统中基于非合作博弈的调度算法[J]. 湖南大学学报(自然科学版),2016,43(10):139-147.

[12] LIU Yu, XIANG Dongyang, ZHENG Chundi. Scheduling and optimizing algorithm for parallel tasks in heterogeneous distributed computing systems[J]. Systems Engineering and Electronics, 2016, 38(2): 332-338. (in Chinese)
柳玉,向东阳,郑春弟. 面向异构分布式计算环境的并行任务调度优化方法[J]. 系统工程与电子技术,2016,38(2):332-338.

[13] BERNSTEIN P A. Middleware: a model for distributed system services [J]. Communications of the ACM, 1996, 39(2): 86-98.

[14] JIA Guojun. Design of transaction middleware system based on three-tiered architectures [J]. Computer Engineering, 2004, 30(14): 79-80. (in Chinese)
贾郭军. 基于三层结构交易中间件系统设计[J]. 计算机工程,2004,30(14):79-80.

[15] LIN Peisheng, WANG Yijun, XUE Zhi. Task scheduling algorithm and protocol for distributed rapid port scanning[J]. Communications Technology, 2017, 50(12): 2787-2793. (in Chinese)
林培胜,王轶骏,薛质. 分布式快速端口扫描的任务调度算法与协议研究[J]. 通信技术,2017,50(12):2787-2793.

[16] BANAVAR G, CHANDRA T D, STROM R E, et al. A case for message oriented middleware[C]//Proceedings of International Symposium on Distributed Computing. New York, USA: ACM Press, 1999: 1-16.

[17] PIETZUCH P, BHOLA S. Congestion control in a reliable scalable message-oriented middleware [EB/OL]. [2019-08-10]. https://xueshu.baidu.com/usercenter/paper/show?paperid=ff6761f3b16079e4a0be643688772b22&site=xueshu_se&hitarticle=1.

[18] LI Jing. Research on scanning task scheduling in distributed network security vulnerability scanning system[D]. Baoding: North China Electric Power University, 2009. (in Chinese)
李静. 分布式网络安全漏洞扫描系统中扫描任务调度的研究[D]. 保定:华北电力大学,2009.

[19] JACKSON K. OpenStack cloud computing cookbook[M]. Birmingham, UK: Packet Publishing Ltd, 2013.

[20] DONG Yaozu, ZHOU Zhengwei. X86-based system virtual machine development and application [J]. Computer Engineering, 2006, 32(13): 71-73. (in Chinese)
董耀祖,周正伟. 基于 X86 架构的系统虚拟机技术与应用[J]. 计算机工程,2006,32(13):71-73.

[21] BELOGLAZOV A, PIRAGHAJ S F, ALROKAYAN M, et al. Deploying OpenStack on CentOS using the KVM hypervisor and GlusterFS distributed file system[EB/OL]. [2019-08-10]. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=1949C2BA266A86A208D4455A14C70A68?doi=10.1.1.296.6475&rep=rep1&type=pdf.

[22] NASCIMENTO M R, ROTHENBERG C E, SALVADOR M R, et al. Quagflow: partnering quagga with OpenFlow[J]. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 441-442.

[23] SOBELL M G. A practical guide to Ubuntu Linux[M]. [S.l.]: Prentice Hall, 2015.

编辑 刘继娟