Journal of System Simulation

Volume 32 | Issue 8 Article 3

8-13-2020

Multi-objective Topology Mapping Method for Network Emulation

Xiaofeng Wang
School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China;

Chen Yang
School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China;

Guangjie Zhang
School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China;

Jianyu Chen
School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China;

Multi-objective Topology Mapping Method for Network Emulation

Abstract

Abstract: Network emulation is an important support for the verification of new network technology, and effective mapping is the key to the emulation network topology. Considering the multiple resource requirements, a multi-objective topology mapping method (MOTM) for network emulation topology is proposed to realize the effective physical resources utilization. The method *analyzes the resource requirements of nodes and links, assigns corresponding weights*, converts the mapping problem into the graph partitioning problem, divides the graph by multi-level graph partitioning method, and forms a mapping strategy *through remote throughput threshold optimization adjustment. The automatic deployment is implemented based on the mapping strategy*. Experiments show that, compared with the Openstack mapping method and the random mapping method, MOTM reduces the load imbalance index by 66.5% and 95.5%, and the telecommunication overhead index by 69.1% and 65.2%.

Keywords

network emulation, topology mapping, Openstack, multi-objective optimization

Recommended Citation

Wang Xiaofeng, Chen Yang, Zhang Guangjie, Chen Jianyu. Multi-objective Topology Mapping Method for Network Emulation[J]. Journal of System Simulation, 2020, 32(8): 1436-1445.

第 32 卷第 8 期 系统仿真学报© Vol. 32 No. 8
2020 年 8 月 Journal of System Simulation Aug., 2020

面向网络仿真拓扑的多目标优化映射方法

王晓锋，陈阳，张光杰，陈建宇
(江南大学 物联网工程学院，江苏 无锡 214122)

摘要：网络仿真是新型网络技术验证的重要支撑，针对给定的网络仿真拓扑，实现有效映射是其关键。综合考虑多种资源需求，提出了多目标优化映射方法 MOTM (Multi-Objective Topology Mapping Method)，实现物理资源的有效利用。该方法分析网络中节点、链路资源需求，赋予相应权值；将映射问题转化为图划分问题，采用多级图划分方法进行划分，并通过远程吞吐量阈值优化调整；最后,基于映射策略实现了仿真拓扑的自动部署。实验表明，MOTM 相对于 Openstack 映射方法、随机映射方法，负载不均衡指数平均降低 66.5%，95.5%，远程通信开销指数平均降低 69.1%，65.2%。

关键词：网络仿真；拓扑映射；Openstack；多目标优化
中图分类号：TP393 文献标识码：A 文章编号：1004-731X (2020) 08-1436-10
DOI: 10.16182/j.issn1004731x.joss.18-0843

Multi-objective Topology Mapping Method for Network Emulation

Wang Xiaofeng, Chen Yang, Zhang Guangjie, Chen Jianyu
(School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China)

Abstract: Network emulation is an important support for the verification of new network technology, and effective mapping is the key to the emulation network topology. Considering the multiple resource requirements, a multi-objective topology mapping method (MOTM) for network emulation topology is proposed to realize the effective physical resources utilization. The method *analyzes the resource requirements of nodes and links, assigns corresponding weights*, converts the mapping problem into the graph partitioning problem, divides the graph by multi-level graph partitioning method, and forms a mapping strategy *through remote throughput threshold optimization adjustment*. *The automatic deployment is implemented based on the mapping strategy*. Experiments show that, compared with the Openstack mapping method and the random mapping method, MOTM reduces the load imbalance index by 66.5% and 95.5%, and the telecommunication overhead index by 69.1% and 65.2%.

Keywords: network emulation; topology mapping; Openstack; multi-objective optimization

引言

网络仿真技术伴随互联网的飞速发展应运而生，已经成为支撑网络空间安全新技术验证、网络安全试验、攻防对抗演练和网络风险评估的重要手段[1]。当前，云计算、SDN 技术和虚拟化技术的网络仿真成为主流方向，虚拟化技术支撑下的网络仿真代表工作包括：美国的休斯顿大学，实现了虚拟化分布式的网络复现技术，并且对链路性能进行了参数设置[2]；美国空军技术学院利用操作系统虚拟化和全虚拟化技术，实现了大规模、高逼真度网络节点仿真平台[3]；中国科学院

收稿日期：2018-12-18 修回日期：2019-03-04；
基金项目：国家自然科学基金(61672264)，国家重点研发计划(2016YFB0800801)；
作者简介：王晓锋(1978-)，男，江苏无锡，博士，副教授，研究方向为网络安全，网络建模与模拟；陈阳(1995-)，女，吉林松原，硕士生，研究方向为网络仿真。

大学基于Openstack云平台构建了TMSR核能仿真系统，能够按需提供仿真计算资源，实现了仿真系统快速灵活部署和重构[4]。

考虑到网络仿真的规模性问题，多个宿主机的分布式仿真成为了主流。需要将一个网络仿真拓扑划分成若干区域，并由不同的宿主机分别承载。由于仿真网络中节点和链路有相应的资源需求，需要综合 CPU、内存、带宽以及远程吞吐量多种因素，从而提高物理资源利用率和提高仿真性能。为此，本文针对划分的多方面目标，提出一种面向网络仿真拓扑的多目标映射方法(MOTM，Multi-Objective Topology Mapping Method)。

1 相关工作

由于网络仿真拓扑的负载和资源配置情况通常是未知的，映射问题被验证是NP难问题。拓扑映射主要有 2 类方法：

(1) 虚拟网络映射

虚拟网络映射算法已经颇有研究成果，例如基于拓扑势的虚拟网络映射算法[5]、基于链路优先的虚拟网络映射算法[6]等，其目的是充分利用有限的底层网络资源，提高底层网络的收益，然而同一虚拟网络中的节点不能共享同一底层物理节点，其目标重在提高收益开销比[7]，并不适合需要保证逼真度的网络仿真任务。

(2) 面向虚拟化技术的仿真任务映射

以虚拟化为基础的网络仿真是时下主流方向[1]，面向虚拟化技术的仿真网络拓扑映射成为了研究热点，代表性工作包括：DETER 支持虚拟化进行大规模试验的平台，它仅实现了基于 METIS 的简单资源映射机制，并且用户需要手动生成从虚拟节点到物理机的映射[8]。美国普渡大学的 Yao 等[9]提出了简易映射方法EasyScale，用METIS和动态规划结合，实现大规模网络安全实验的简易映射，但是把链路带宽设置相同数值，对带宽参数考虑不全面。Ricci 等[10]提出了基于模拟退火算法实现任务映射的算法，该方法考虑了虚拟和物理资源的灵活性，能够找到接近最优解的结果，但是没有把链路参数考虑成定量指数。邱长伶等[11]提出了一种结合轻量级和全虚拟化方法，对大规模网络拓扑进行划分的 TMMI 算法，可以实现弹性映射，负载均衡，但是该方法仅分成边缘节点和非边缘节点，对非边缘节点看成是相同配置，欠缺对仿真网络中各种不同节点具有不同资源需求的情况。基于Openstack的网络仿真最具有代表性，Li等[12]基于Openstack开发了分布式同步仿真模块，实现了同步、精确和实时的网络仿真。但 Openstack 云平台自身的映射方法每次只能处理一个虚拟机请求，不能根据资源请求进行统一的调度，回溯，因此会造成映射结果不准确，并且在映射过程中没有考虑到吞吐量对网络仿真的影响，所以不适合直接应用于网络仿真[13]。

上述方案，采用不同方法实现了对虚拟网络任务的映射，同时提升了网络仿真任务的物理资源的利用率，使网络仿真任务更好地进行，但是吞吐量对系统进行仿真实验的影响很大，以上方法没有考虑到网络仿真全面综合的优化目标；并且对于每个节点，虚拟机分配不同资源可以更好地利用底层资源，提高资源利用率；同时大多数方法只是做到了对网络拓扑有一个划分结果，在对物理资源和虚拟资源之间的映射需要手工部署，手工映射给实验增加了很大的难度和代价。

本文针对上述问题，提出 MOTM 方法，赋予核心程度不同的虚拟机不同权重，综合考虑网络仿真底层物理资源各个方面的优化目标，同时以最小化远程吞吐量为划分目标，完成多目标拓扑划分。通过 MOTM 方法划分后，将划分结果转换为网络拓扑的自动部署脚本，可以和Openstack无缝集成，从而实现更灵活的映射。

2 多目标拓扑映射问题描述

2.1 基于云计算的拓扑映射因素分析

2.1.1 网络拓扑映射影响因素

为有效地利用底层物理资源，虚拟网络映射

需要考虑到多种因素，在满足资源约束条件的前提下，充分地利用物理网络资源[14]。

影响网络映射的因素主要有节点资源需求和链路资源需求。网络仿真对节点资源需求包括 CPU、内存和磁盘容量大小。在实际应用中，由于磁盘空间一般较充足并且可扩展，为此需要重点考虑 CPU 和内存。

网络仿真对链路资源需求主要针对可用带宽资源，可用带宽应满足宿主机内部吞吐量以及远程吞吐量大小。计算节点内部带宽限制可容纳的虚拟机个数，远程链路带宽限制跨宿主机节点间的链路条数。

2.1.2 吞吐量对拓扑划分的影响

针对宿主机吞吐量问题，本文做了大量的测试实验，来确定远程链路对宿主机内部吞吐量的影响。测试实验均基于 Openstack 平台，配置为 1 个控制节点，1 个网络节点，2 个计算节点。计算节点均为服务器 R830，CPU 类型为 2 个 Intel E5620 处理器，内存为 64 GB。每个虚拟机分配 1 个 CPU，4 GB 内存资源；操作系统为 Ubuntu16.04。

(1) 宿主机内吞吐量上限测试

图 1 为用 netperf 测试在同一宿主机内不同跳数的同一时刻吞吐量上限，宿主机内的吞吐量随着跳数增加而上升，到一定跳数会出现上限，选取达到上限的 12 跳进行进一步的测试。

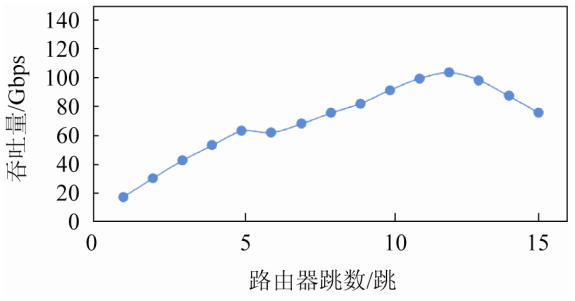

图 1 虚拟路由器吞吐量对比图
Fig. 1 Virtual router throughput comparison chart

(2) 跨宿主机间吞吐量开销测试

图 2 为在宿主机内的吞吐量达到上限时，远程链路吞吐量大小对宿主机内部吞吐量大小影响的拟合图，取不同大小的远程链路吞吐量，测出该远程吞吐量下宿主机内吞吐量的值，每组进行十组测试并取实验结果的平均值进行线性拟合，分析远程吞吐量大小对内部吞吐量的影响。

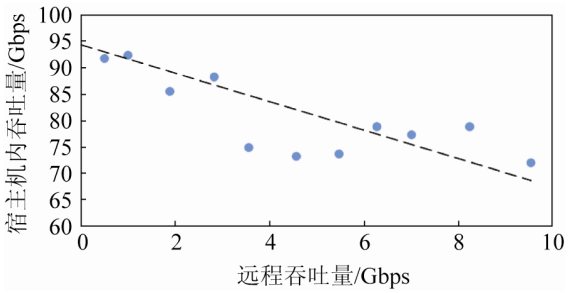

图 2 远程流量对宿主机内流量影响的平均值拟合图
Fig. 2 Average fit graph for impact of remote traffic on host traffic

设远程吞吐量大小为 *ExtTp*，宿主机内部吞吐量大小为 *IntTp*，宿主机内吞吐量上限为 *TpCap*，则有：

$$\alpha ExtTp + IntTp \leqslant TpCap \tag{1}$$

通过对每组实验的平均值进行拟合得出远程吞吐量大小对内部吞吐量影响的系数 α=–2.706 3。

对实验数据进行分析，可得出拓扑划分时需要考虑到的吞吐量限制：(1) 宿主机内的吞吐量是有上限的，所以划分时应该考虑到宿主机吞吐量阈值；(2) 随着远程吞吐量增大，宿主机内吞吐量呈线性减少，使得仿真性能受到影响，所以减小远程吞吐量对仿真性能有很大的提高。

Openstack 云平台在对网络仿真拓扑进行部署时，只考虑了不同服务器之间 CPU、内存和磁盘的大小的分布均衡，将网络拓扑中的虚拟机看成是单独的个体进行映射，没有考虑到网络仿真的整体性，忽视了吞吐量这一重要因素。

2.2 网络仿真拓扑映射目标分析

通过上述对影响网络拓扑映射因素的分析，得出在映射时要进行优化的目标。映射应该考虑到 CPU、内存和吞吐量 3 个基本物理资源，提高资源利用率和负载均衡程度是映射的首要优化目标。与此同时，由于远程吞吐量的大小会很大程度上影响到仿真性能，所以映射时需要尽可能减

少远程链路间吞吐量大小，因此应该将最小化远程链路吞吐量大小作为优化目标。

3 多目标优化映射方法

基于 2.2 节的分析，本文提出了 MOTM 方法，该方法基于多级图划分方法(METIS)来进行划分。METIS 划分的目标是：各子图点权值和满足一定均衡的情况下，被划分边权值和最小[15]，与拓扑划分的需求高度一致，所以使用 METIS 图划分作为本文划分的基础算法，并对对应的点边权进行合理的制定。其整个架构如图 3 所示。

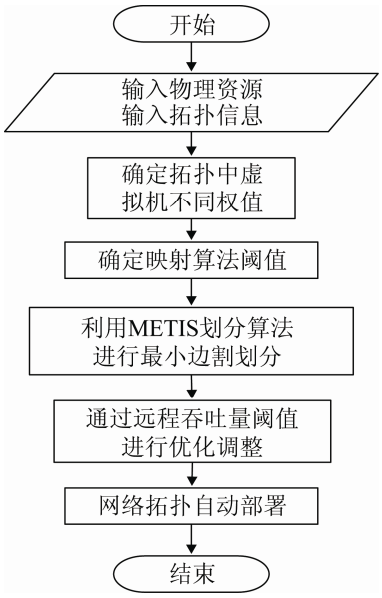

图 3 MOTM 整体流程
Fig. 3 MOTM Overall process

MOTM 方法基于 Openstack 云平台进行实验，使用全虚拟化方式进行部署，因为KVM是硬件级别的虚拟化技术，性能稳定，相比较而言以 Docker 为代表的轻量级虚拟化主要面向应用，更适合快速部署和大规模构建，所以选用KVM来保证网络任务的可用性。在网络实验中，每个节点所承载的任务量是有区别的，应考虑到不同节点核心程度，给予不同资源配置参数和链路带宽参数，而不是统一分配相同配置虚拟机，MOTM 通过拓扑中每个节点连接的节点个数的不同，即节点的度不同，将其分类成不同配置，从而达到使虚拟机差异化的目的。兼顾计算节点利用率和远程通信量 2 个目标，利用多级图划分方法进行划分后，自动生成部署脚本，实现划分、映射、部署的总体流程。

映射总体流程由 3 个部分组成：(1) 仿真拓扑参数确定；(2) 多级图划分和优化；(3) 自动部署。

仿真拓扑参数确定采用虚拟机分类算法，通过计算拓扑节点中度的不同，将带宽指数分配给拓扑中的每个节点。在划分时，按照带宽指数来分配不同的 CPU 指数和内存指数，更多的资源被分配给具有高指数的节点。

多级图划分和优化算法首先采用 METIS 划分，根据仿真拓扑参数确定算法，设置 3 个点权一个边权，综合考虑多方面的影响，使划分负载均衡；然后对划分结果进行远程链路阈值判断，从而进行调整。

MOTM 生成映射结果后，自动部署模块读取映射结果，调用自动部署程序，实现将实验拓扑自动搭建在 Openstack 上。

3.1 仿真拓扑参数确定

仿真拓扑通过利用分类算法来确定虚拟机参数。由于 K-means 聚类方法具有简单方便性，本文使用此算法对拓扑节点进行分类。K-means 是一种非监督实时、基于划分的聚类方法。将数据划分为预定的 K 类，迭代逐次更新到聚类的值[16]。

根据拓扑中每个节点的度的不同，通过 K-means 算法进行分类。定义虚拟机资源表示为：

$$Rvm = < cpu_i, mem_i, disk_i > \tag{2}$$

通过分类结果 c 的取值，对应匹配虚拟机 3 个不同配置等级：

$$(cpu_i, mem_i, disk_i) = \begin{cases} (1\text{ vcpu, }1\text{ GB, }5\text{ GB}), & c=1 \\ (2\text{ vcpu, }2\text{ GB, }20\text{ GB}), & c=2 \\ (3\text{ vcpu, }4\text{ GB, }40\text{ GB}), & c=3 \end{cases} \tag{3}$$

根据链路两端虚拟机的大小，设定链路带宽大小，给予连接配置大的虚拟机的链路更高的带宽。

算法流程如下：

DOI: 10.16182/j.issn1004731x.joss.18-0843

(1) 读取网络拓扑的节点和链路信息，计算出节点的度；

(2) 设置预定类数 n_clusters=3，最大迭代次数 max_iter=100；

(3) 根据虚拟机分类的结果 c (c=0, 1, 2)，匹配 small, medium, large 三种配置大小，同时对 $(cpu_i, mem_i, disk_i)$ 赋值；

(4) 考虑链路两边的虚拟机的配置不同，配置越大，一般实际网络任务中虚拟机的吞吐量越大，从而给链路带宽 $throughput_i$ 赋值；

(5) 将虚拟机 $Rvm = <cpu_i, mem_i, disk_i>$ 和链路带宽 $throughput_i$ 的赋值情况进行存储记录，供下一步划分使用。

3.2 多级图划分和优化

多级图划分和优化的总体流程如图 4 所示。

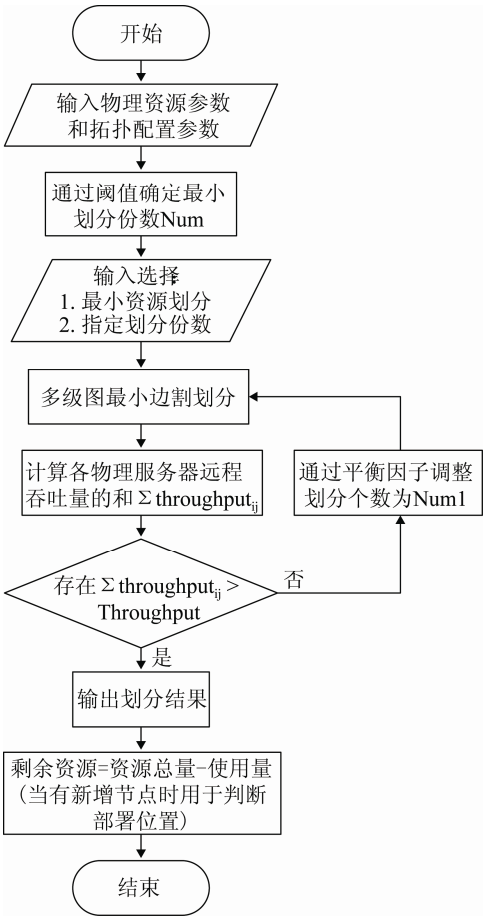

图 4 多级图划分和优化的总体流程图

Fig. 4 Overall flow chart for multi-level graph partitioning and optimization

具体算法如下：

(1) 输入和输出

1. 读取物理计算节点的资源定义为：

$$P = <CPU, MEMORY, VOLUME> \tag{4}$$

2. 读取通过分类算法给定权值后的虚拟拓扑配置参数信息；

3. 生成划分时可以选择：a. 最小资源划分(计算出来的所需计算节点数量)，b. 指定划分数量(最多有多少计算节点可用)平衡系统中的主机的计算资源，提高应用性能；

4. 划分结果输出文件处理成每个宿主机包含的节点编号，存储每个虚拟机对应的宿主机编号。

(2) 映射划分过程

由于大多数测试平台仅包含几组同构物理机器，所以资源按着同构进行处理；映射生成算法根据现有的虚拟机权值参数，生成指定的 METIS 输入文件，生成一组合理映射。

1. 拓扑的节点数为 N，链路数为 M。遍历拓扑，得到由 2.1 节中确定的每个节点的 $Rvm = <cpu_i, mem_i, disk_i>$ 以及每条链路带宽 $throughput_i$；

2. 通过阈值确定所需的最小划分份数，首先仅按照部署节点所需逻辑 CPU 个数，来确定理论所需计算节点的个数 C：

$$C = \frac{\sum_{i=0}^{N-1} cpu_i}{CPU}, C\text{取整数} \tag{5}$$

3. 计算仅按照部署节点所需的内存大小，来确定理论所需计算节点个数 M：

$$M = \frac{\sum_{i=0}^{N-1} mem_i}{MEMORY}, M\text{取整数} \tag{6}$$

4. 节点所连接的每条链路带宽 $throughput_j$，设定连接链路条数为 m，令 $sumFIi$ 设定为一个节点的带宽指数的和，对每个节点的带宽指数设定为 $sumFI_i(throughput_j)$；计算仅按照部署节点所需的吞吐量大小，确定理论所需的计算节点个数 V：

$$V=\frac{\sum_{i=0}^{N-1} sumFI_i(throughput_j)}{VOLUME\bullet 2},V\text{取整数} \tag{7}$$

5. 通过取以上 3 个阈值最大值，得出同时满足 CPU、内存和吞吐量 3 个参数值的最小划分份数 *Num*：

$$Num=\max\{C,M,V\} \tag{8}$$

6. 生成 metis 划分输入文件，设置 3 个点权(每个节点的 CPU、内存和吞吐量总和)和一个边权(链路吞吐量)，按照划分份数(*Num* 或指定划分数量)，用 METIS 图分割库，以最小化分区之间的虚拟链接数来划分；

7. 用矩阵 ***A*[*N*-1][*N*-1]**存储拓扑中对应节点间链路吞吐量；

8. 建立拓扑节点与物理节点对应的矩阵 ***B*[*Num*][*N*-1]**存储划分结果，行坐标为物理节点编号，列坐标为对应拓扑节点的编号；

(3) 映射调整部分

1. 设定每台服务器上的节点集合为 R_k(k=1, 2,..., *Num*)，节点 i 和节点 j 之间的链路吞吐量为 $Throughput_{ij}$，根据划分结果矩阵 ***B***，得出第 k 个服务器远程吞吐量：

$$remoteT_k=\sum_{(i\in R_k\wedge j\notin R_k)} Throughput_{ij}^{k} \tag{9}$$

2. 设定划分结果的区域间远程吞吐量的最大值为 T，则 T 的值为：

$$T=\max(remoteT_k) \tag{10}$$

3. T 与远程吞吐量阈值 *Throughput* 进行比较，如果超过则通过调整因子对划分初始个数 *Num* 进行调整：

$\exists(T>Throughput)$；

$$Num=Num+(e-1)^{\frac{T-Throughput}{Throughput}} \tag{11}$$

4. 用修正过的划分份数重新划分计算，重复(2)中 6，7，8 和(3)中 1，2，3 步骤，直到满足各方面阈值；

5. 根据生成的满足条件的矩阵结果 ***B***，生成划分结果文件，将节点编号与对应服务器编号对应；

6. 计算各服务器的剩余资源，存储剩余资源情况，当需要新增虚拟机节点时，可以快速确定虚拟机放置位置。

4 基于映射策略的拓扑自动部署

4.1 自动部署方法

一旦确定了从实验拓扑到物理资源的映射，在手动在物理机上建立实验是耗时的。配置过程不仅需要在底层扩展技术方面强大的专业知识，而且还涉及大量的配置脚本。所以自动化配置的实现是十分有意义的。

基于全虚拟化技术，使用 KVM 虚拟机将虚拟节点部署在服务器上。自动部署程序通过研究 Openstack 的 python API，封装常用的 Python SDK 功能，分别是认证功能 auth，镜像功能 glance，网络功能 neutron，计算功能 nova 以及端口设置功能 port，供自动部署调用；然后通过配置文件描述网络拓扑结构，对配置文件进行解析得到所需配置参数；最后调用接口和配置参数，完成自动化部署。

自动部署程序的接口调用关系如图 5 所示。

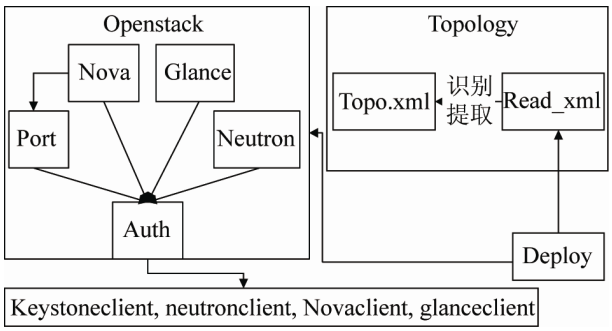

图 5 自动部署接口调用关系

Fig. 5 Call relationship of an automatically deployed interface

4.2 自动部署具体流程

根据划分过程生成的映射结果，生成拓扑描述文件，解析参数信息，再调用部署接口进行自动化部署。创建工作流如图 6 所示。

DOI: 10.16182/j.issn1004731x.joss.18-0843

第 32 卷第 8 期 系统仿真学报 Vol. 32 No. 8
2020 年 8 月 Journal of System Simulation Aug., 2020

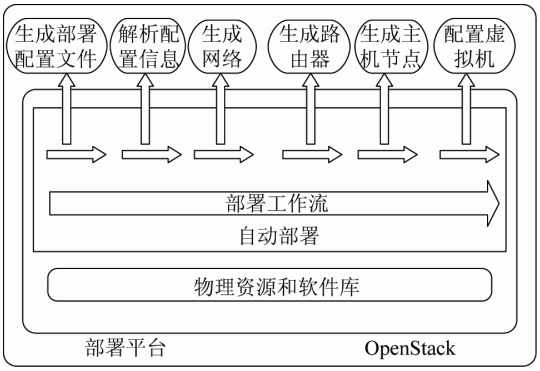

图 6 自动部署工作流
Fig. 6 Automated deployment workflow

5 实验验证与分析

5.1 网络仿真拓扑映射效果评估方法

在网络仿真中，与拓扑划分的有关因素有负载均衡因素、通信开销因素以及模拟应用因素。其中负载均衡和远程通信开销为并行仿真实验自带的固有属性，无论何时均应该保证其等于最优值[17]。

定义 1 负载不均衡指数 LIBI 用于评价物理集群的负载均衡程度，当虚拟网络拓扑被划分的越均衡时，负载不均衡指数数值将越小。负载均衡是指物理节点和物理链路之间的负载处于相对均衡的状态，不出现物理节点或链路负载过大等现象。综上，负载不均衡指数用公式(12)描述：

$$LIBI = \frac{1}{n}\sum_{i=1}^{n}\left[(\overline{cpu}-cpu_i)^2+(\overline{mem}-mem_i)^2+\frac{1}{10}\left(\frac{\overline{vol}-vol_i}{1000}\right)^2\right] \tag{12}$$

式中：n 为划分所需物理计算集群规模；$\overline{cpu}$ 为所有物理计算节点平均占用 vcpu 个数；cpu_i 为每个计算节点被占用的 vcpu 个数；$\overline{mem}$ 为物理计算节点平均内存占用量；mem_i 为每个计算节点内存占用量；$\overline{vol}$ 为物理计算节点平均吞吐量占用量；vol_i 为每个计算节点吞吐量占用量。由负载不均衡指数判断负载均衡度，数值越小表示负载均衡度越优。

定义 2 远程通信开销指数 RTI 用于评价不同物理计算节点之间远程链路通信量大小的情况。虚拟机跨物理节点通信时，需要通过远程链路进行数据传输，划分时要保证远程通信量总和尽可能的小。远程通信开销指数用如下公式描述：

$$RTI = \sum_{i=1}^{n} rthrough_i \tag{13}$$

式中：n 为物理集群个数；$rthrough_i$ 为每个物理节点远程通信量值；RTI 的数值越小，远程通信量越少，则划分结果越优。

下文通过对创建的拓扑进行 3 组方法的对比实验结果，分析不同方法的 LIBI 和 RTI 值，来评价 MOTM 算法的效果。

5.2 实验的软硬件环境

为了验证本文提出的映射方法和自动部署功能的有效性，构建了一套 Openstack 平台，配置为 1 个控制节点，1 个网络节点，4 个计算节点；操作系统为 Ubuntu16.04。计算节点均为服务器 R830 ，CPU 类型为 2 个 Intel E5620 处理器，内存为 64 GB，计算节点自身吞吐量为 11 Gbps。

5.3 拓扑映射性能实验分析

5.3.1 负载不均衡指数实验分析

为了验证本文方法的有效性，采用 GT-ITM 的 Transit-Stub(TS)模型，分别生成 60，80 和 100 个节点的 3 组网络拓扑，对 LIBI，RTI 指标进行验证。并且与随机划分算法和 Openstack 映射方法进行了比较分析。

虚拟网络拓扑通过 MOTM 算法、Openstack 映射方法以及随机划分算法划分后，由公式和实验数值得到 3 组实验对应的 LIBI 值如表 1 所示。负载不均衡指数对比如图 7 所示。

根据实验结果可知，MOTM 算法和 Openstack 映射方法得到的负载不均衡指数远小于用随机算法的得到的数值，并且从总体来看 MOTM 得到的均衡指数值优于 Openstack 映射方法得到的值，通过对 3 组实验结果平均值计算出 MOTM 的 LIBI 值相对于 Openstack 映射方法降低 66.5%，相对于随机方法降低了 95.5%，所以 MOTM 算法可以更好的保证划分后物理集群的负载均衡，减少单个物理服务器过载现象。

表 1 虚拟网络拓扑各性能对比
Tab. 1 Virtual network topology performance comparison

性能	算法	60 节点拓扑	80 节点拓扑	100 节点拓扑
远程通信开销指数 (10^6 bit/sec)	MOTM 算法	44 000	38 000	112 000
	随机算法	107 000	178 000	273 000
	Openstack 映射方法	127 000	202 000	299 000
负载不均衡指数	MOTM 算法	1.13	0.70	1.42
	随机算法	21.29	24.09	26.85
	Openstack 映射方法	7.50	1.25	0.93

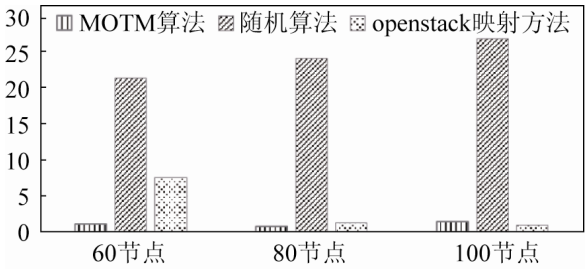

图 7 负载不均衡指数对比图
Fig. 7 Load imbalance index comparison chart

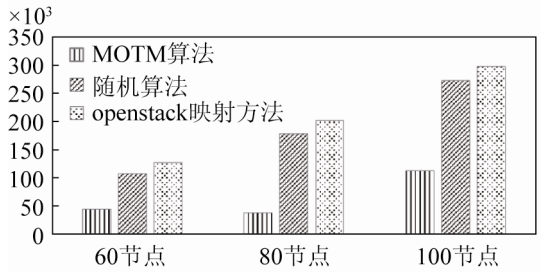

图 9 远程通信开销指数对比图
Fig. 9 Remote throughput overhead index comparison chart

5.3.2 远程通信开销指数实验分析

80 节点虚拟网络拓扑通过 3 种划分方法后，每个物理计算节点的远程通信量对比图如图 8 所示。

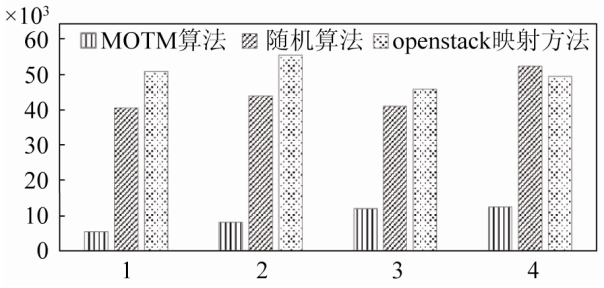

图 8 80 节点拓扑各计算节点远程通信量对比图
Fig. 8 Comparison of remote communication volume of each computing node in 80-node topology

从图 8 中可以看出，用 MOTM 算法得到的每个节点的远程通信量都远小于其他 2 种方法，并且分布较为均衡。

由公式和实验数值得到 3 组实验对应的 RTI 值如表 1 所示。远程通信开销指数对比图见图 9。

实验网络拓扑性能对比信息如表 1 所示，实现结果表明，通过 MOTM 算法划分网络拓扑，可以降低不同物理节点之间的远程通信量大小，其值远小于通过随机划分和 Openstack 映射方法得到的远程通信量指数值。通过对 3 组实验结果平均值可计算出 MOTM 的 RTI 值相对于 Openstack 映射方法降低了 69.1%，相对于随机方法降低了 65.2%，由此可见，MOTM 算法可以更好地划分实验拓扑，使得远程通信量更小。

5.4 自动部署实验

通过自动部署程序，将网络仿真所需拓扑自动化部署在 Openstack 平台上，通过 dashboard 查看到 80 节点拓扑部署结果如图 10 所示。

借助自动部署功能，用户可以将资源映射快速的转换到 Openstack 平台上，自动化搭建实验拓扑，从而减少了长时间的手工创建过程。

DOI: 10.16182/j.issn1004731x.joss.18-0843

第 32 卷第 8 期 系统仿真学报 Vol. 32 No. 8
2020 年 8 月 Journal of System Simulation Aug., 2020

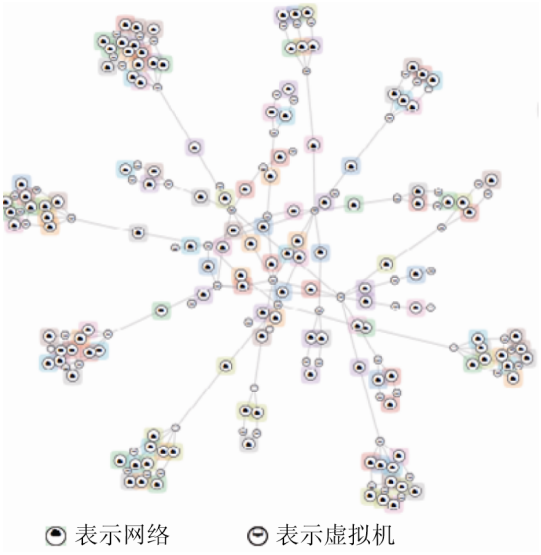

图 10 80 节点虚拟拓扑 Openstack 部署图
Fig. 10 80-node virtual topology deployment diagram on Openstack

6 结论

本文中，我们提出了一个新的算法 MOTM，用于映射配置网络实验。该方法通过对输入实验拓扑进行分析，给虚拟机不同的权值，考虑到远程吞吐量影响，对资源进行合理利用，并通过全虚拟化技术进行拓扑映射。

使用 MOTM 方法可以完成实验任务的自动映射和部署，操作方法简易方便。通过对比实验结果表明，MOTM 的性能优于 Openstack 映射方法和直接应用 METIS 进行随机划分 2 种方式。MOTM 可以划分各种网络实验。下一步工作主要研究在异构情况下的划分以及对自动化部署多类型节点互联互通。

参考文献:

[1] 方滨兴, 贾焰, 李爱平, 等. 网络空间靶场技术研究[J]. 信息安全学报, 2016, 1(3): 1-9.
Fang Binxing, Jia Yan, Li Aiping, et al. Research on Network Space Shooting Range Technology[J]. Journal of Cyber Security, 2016, 1(3): 1-9.

[2] Dutta A, Gnawali O. Large-scale network protocol emulation on commodity cloud[C].2014 IEEE Global Communications Conference. Austin: IEEE, 2014: 1114-1119.

[3] Andel T R, Stewart K E, Humphries J W. Using Virtualization for Cyber Security Education and Experimentation[C]. 14th Colloquium for Information Systems Security Education. Baltimore: CISSE, 2010: 130-136.

[4] 何越. 基于云计算技术的 TMSR 仿真平台设计及性能分析[D]. 上海: 中国科学院大学(中国科学院上海应用物理研究所), 2018.
He Yue. Design and performance analysis of TMSR simulation platform based on cloud computing technology[D]. Shanghai: University of Chinese Academy of Sciences (Shanghai Institute of Applied Physics, Chinese Academy of Sciences), 2018.

[5] 刘新波, 王布宏, 杨智显, 等. 一种基于拓扑势的虚拟网络映射算法[J]. 电子与信息学报, 2018, 40(7): 1684-1690.
Liu Xinbo, Wang Buhong, Yang Zhixian, et al. A Virtual Network Mapping Algorithm Based on Topological Potential[J]. Journal of Electronics & Information Technology, 2018, 40(7): 1684-1690.

[6] 熊文成, 王颖, 邱雪松, 等. 基于链路优先的快速协同虚拟网络映射算法[J]. 通信学报, 2015(3): 249-257.
Xiong Wencheng, Wang Ying, Qiu Xuesong, et al. Fast cooperative virtual network mapping algorithm based on link priority[J]. Journal of Communications, 2015(3): 249-257.

[7] 程祥, 张忠宝, 苏森, 等. 虚拟网络映射问题研究综述[J]. 通信学报, 2011, 32(10): 143-151.
Cheng Xiang, Zhang Zhongbao, Su Sen, et al. Review of virtual network mapping problems[J]. Journal of Communications, 2011, 32(10): 143-151.

[8] DETER Team. Building apparatus for multi-resolution networking experiment using containers[R]. DeterLab, Tech. Rep. ISI-TR-683, 2011.

[9] Yao W M, Fahmy S, Zhu J. EasyScale: Easy mapping for large-scale network security experiments[C]. 2013 IEEE Communications and Network Security (CNS): IEEE, 2014: 269-27.

[10] Ricci R, Alfeld C, Lepreau J. A solver for the network testbed mapping problem[J]. ACM SIGCOMM Computer Communication Review (S0146-4833), 2003, 33(2): 65-81.

[11] 刘渊, 邱常伶, 王晓锋, 等. 面向多尺度融合网络仿真的拓扑映射方法研究[J]. 系统仿真学报, 2019, 31(10): 2030-2041.
Liu Yuan, Qiu Changling, Wang Xiaofeng, et al. Research on Topological Mapping Method for Multiscale Fusion

Network Simulation[J]. Journal of System Simulation, 2019, 31(10): 2030-2041.

[12] Li H F, Zhou H C, Zhang H K, et al. Emustack: An openstack-based dtn network emulation platform (extended version)[J]. Mobile Information Systems (S1574-017X), 2016.

[13] 张莉莉. 基于 OpenStack 的虚拟机资源调度关键技术研究[D]. 北京: 北京邮电大学, 2015.
Zhang Lili. Research on Key Technologies of Virtual Machine Resource Scheduling Based on OpenStack[D]. Beijing: Beijing University of Posts and Telecommunications, 2015.

[14] 蔡志平, 刘强, 吕品, 等. 虚拟网络映射模型及其优化算法[J]. 软件学报, 2012, 23(4): 864-877.
Cai Zhiping, Liu Qiang, Lü Pin, et al. Virtual Network Mapping Model and Its Optimization Algorithm[J]. Journal of Software, 2012, 23(4): 864-877.

[15] 王晓锋, 方滨兴, 云晓春, 等. 并行网络蠕虫模拟中任务优化划分的研究[J]. 计算机学报, 2006, 29(8): 1367-1374.
Wang Xiaofeng, Fang Binxing, Yun Xiaochun, et al. Research on task optimization partitioning in parallel network worm simulation[J]. Chinese Journal of Computers, 2006, 29(8): 1367-1374.

[16] 李斌, 王劲松, 黄玮. 一种大数据环境下的新聚类算法[J]. 计算机科学, 2015, 42(12): 247-250.
Li Bin, Wang Jinsong, Huang Wei. A New Clustering Algorithm in Big Data Environment[J]. Computer Science, 2015, 42(12): 247-250.

[17] 葛文堂, 张兆心, 李斌. 基于模拟运行时间的拓扑划分评价模型[J]. 通信学报, 2013, 34(6): 122-127.
Ge Wentang, Zhang Zhaoxin, Li Bin. Topological division evaluation model based on simulation running time[J]. Journal of Communications, 2013, 34(6): 122-127.

DOI: 10.16182/j.issn1004731x.joss.18-0843