第 38 卷第 Z2 期 通 信 学 报 Vol.38 No.Z2
2017 年 11 月 Journal on Communications November 2017
doi:10.11959/j.issn.1000-436x.2017279

基于超算环境的面向多租户的轻量级虚拟 HPC 集群的设计与实现

谭郁松[1]，李荣振[1]，吴庆波[1]，张建锋[1]，张尧学[1,2]
(1. 国防科技大学计算机学院，湖南 长沙 410073；2. 中南大学信息科学与工程学院，湖南 长沙 410083)

摘 要：为了给用户提供按需使用的 HPC 服务并解决用户应用部署的软件依赖性问题，在不破坏现有超大规模高性能集群管理的前提下，基于 Fat-Tree 网络拓扑和虚拟集群模型，通过对部分资源进行云化管理，设计并实现了一种轻量级的 HPC 集群交付模式。从而为面向租户需求的高性能应用提供一种云化的 HPC 租用服务，并解决了 HPC 应用软件栈的僵化问题，以易用的方式为更多的 HPC 租户提供服务。实验结果表明，该方法以近似物理节点的性能使 HPC 具备了云的按需使用的特性，并具有较为理想的植入效率。

关键词：轻量级虚拟 HPC；虚拟集群模型；胖树；多租户
中图分类号：TP338 **文献标识码**：A

Design and implementation of multi-tenant oriented lightweight virtual HPC cluster based on supercomputing environment

TAN Yu-song[1], LI Rong-zhen[1], WU Qing-bo[1], ZHANG Jian-feng[1], ZHANG Yao-xue[1,2]
(1. College of Computer, National University of Defense Technology, Changsha 410073 China;
2. School of Information Science and Engineering, Central South University, Changsha 410083 China)

Abstract: In order to deliver on-demand HPC services for multi-tenants and solve software dependency problems which rigidly restrict the usage of HPC applications. On the premise of not destroying the existing large scale high performance cluster management, a lightweight HPC cluster delivery model was designed and implement with the Fat-Tree network topology and the virtual HPC cluster model. So as to supply a cloud based HPC rental service for the high performance applications of the multi-tenants and solve the rigid dependable problem of HPC application software stack for more HPC tenants in an easy way, the experimental results show that the proposed method is similar to the performance of the physical nodes, so that the HPC has the on-demand characteristics of the cloud and has a better embedding efficiency.

Key words: light virtual HPC cluster, virtual cluster model, Fat-Tree, multi-tenant

1 引言

随着云计算的快速发展，基于云服务的 HPC 也受到更多的关注。特别是在如今的大数据时代，不仅在科学与学术界，各个领域甚至个人用户都可能有需求利用高性能计算来解决问题。在过去的这些年里，HPC 已退居学术和科学领域，作为一种特有的方法，解决一些最具挑战性的问题。比如，高性能计算在生物信息，基因测试，核动力模拟和天气模拟等多个科学计算领域表现突出。然而在应用需求日益增加的大数据时代，高性能计算集群的交付模式上仍存在诸多问题。在实际的使用过程中，用户应用的软件分组依赖是高性能计算管理中经常遇到的问题，而且在资源的分配上通常采用资源静态分配的方式，无法及时地为用户提供按需的高效资源交付模式。同时，云计算模式则以一种有效的方式将传统的 HPC 应用迁移到云上，在云化的环境下，不仅可以有效地提高 HPC 的交付效率而且使 HPC 的使用方式更加灵活。

HPC 作为一种成熟的并行计算模式，在解决大规模分布式并行计算问题上具有重要作用。然而，通常的高性能计算是一种本地的服务模式，用户需要到达超算中心，然后自己部署自己的并行应用系统或采用类似 VPN 的方式远程操作。无法实现按

收稿日期：2017-09-10

需使用的高效交付模式，从而在某种程度上限制了高性能计算应用的普及和发展。另一方面，用户原有的开发环境并不一定和超算中心的软件版本一致，导致用户难以顺利部署自己的应用到高性能计算集群系统上，这无形中增加了用户的工作量和经济开销。而且由于软件版本、OS 以及应用层接口等问题，导致需要花费大量人力解决软件系统自身的问题。

软件作为一种商品在市场上流通比硬件更为便利，并通过 Web 或信息流的形式传送给终端用户，这种商业模式逐渐成熟，因此采用云服务模式交付高性能计算服务是一种可行的方案。同时，轻量级虚拟化技术以极低的性能开销换得巨大的软件部署灵活性，非常有利于解决僵化的高性能计算软件栈、应用软件的依赖关系以及版本兼容性问题。因此，为了有效提供按需使用的 HPC 交付模式并改进系统的易用性，本文提出了一种基于超算中心胖树（Fat-Tree）网络拓扑和虚拟集群模型设计并实现面向多租户的 HPC 服务模式——轻量级虚拟 HPC 集群，它以很小的系统开销解决了高性能计算领域中上述问题。和其他类似工作[1~3]不同的是，本文将超算中心基础设施云化管理，而非基于商业云构建 HPC 虚拟集群应用，这样可以充分发挥超算资源高性能优势，并提供商业化易用的 HPC 服务。

本文的主要贡献如下。

1) 设计并实现了一种面向多租户的基于 docker 的轻量级虚拟集群。

2) 在不破坏现有高性能计算资源管理系统的前提下，提出一种新的面向超算中心多样化应用的有效方法，并结合云的多租户和轻量级虚拟化来解决应用软件依赖问题。

3) 基于 Fat-Tree 和虚拟集群模型设计了一种新的系统结构，既保持了原有系统的完整性又可以提供按需付费的云化虚拟 HPC 集群交付模式。

4) 进行了性能测试和集群化部署测试，验证本设计的可行性。

2　相关工作

超级计算机在超大规模并行任务处理上具有不可撼动的地位，云计算的发展不是取代超级计算集群，而是在资源交付模式与增加应用多样性等方面为超级计算的发展注入新的动力。高性能集群系统的应用部署复杂及访问方式受限和日益增多的用户需求之间矛盾突出，加上僵化的网络结构和软件栈，以及其他应用所需系统环境的兼容性问题，造成高性能计算市场推广缓慢从而在某种程度上限制了 HPC 的发展。在一些研究中，已有研究者将云计算模式引入高性能计算领域，这种按需付费的使用模式可以很好地解决 HPC 领域面临的这些问题。AbdelBaky 和 Parashar 等[1]在促使高性能计算即服务一文中指出云计算可以作为一种有效的原型框架，并提出将 Blue Gene 系统云化管理来运行 HPC 应用。Niu 等[2]提出一种基于资源预留和动态分配的半弹性虚拟集群，通过工作负载聚合来提高资源利用率进而节约成本。Xavier[3]基于云平台采用的 KVM 和 docker 技术，从占用磁盘空间大小、启动速度和 CPU 性能方面做了分析，凸显 docker 在这几方面优势。Slominski 等[4]在迁移传统 Web 应用到基于 docker 的多租户云上提出了一种迁移传统 Web 应用到基于 docker 的云上的可重用架构。本文从超算中心资源交付模式和应用多样性、易用性角度出发，充分发挥云计算交付模式优势，提出轻量级虚拟 HPC 集群，一种有效的交付 HPC 集群模式，进而为用户提供一种按需使用的高性能集群使用模式，并解决了超算集群系统应用兼容等易用性问题。

在如今的超算建设中，高性能超级计算中心通常拥有巨量的计算节点，如广州超算中心有超过 20 000 个节点，可以满足大量用户应用的需求。但是如此巨大规模的资源，通常只有少量的用户知悉如何申请使用。而云计算的兴起，为其提供了一种新的思路，通过为用户租赁资源的方式，不仅提供了按需使用的交付模式而且解决了 HPC 的软件栈依赖性问题。

现有超算环境一般是由一个巨大的集群构成，可以用于处理超大规模计算任务，虽然从管理者的角度较为方便管理，但资源利用效用比较低，且难以满足用户特定的应用需求。其主要具有以下特点及可能存在的问题。

1) 应用软件单一，节点软件栈是同构的，难以满足用户不同应用的需求。

2) 网络隔离性差，只为用户提供简单的资源分区功能。

3) 动态伸缩性差，其要求所有节点互联互通，即使有部分计算几点不通，也会影响到作业提交的响应。

本文基于虚拟集群模型，采用 Openstack 云服务对现有超算 Fat-Tree 网络拓扑和节点软件栈进行优化设计，实现灵活的网络和节点管理，采用 docker 轻量级虚拟化技术设计并实现一种云上交付 HPC 服务的模式。虚拟集群模型是解决云上提供虚拟集群服务的有效模型，docker 轻量级虚拟化是一种为用户提供 container 服务支撑技术，它具有启动快、资源占用小、支持单节点多容器和用户应用环境快速迁移等特点。同时，为了满足不同用户应用软件差异的问题，解决软件版本依赖问题，docker 可快速实现用户应用软件打包与部署优化。总之，本文的系统架构设计主要具有如下特点。

1) 用户现有应用软件环境快速迁移。
2) 计算资源整合，提高资源利用率。
3) 解决应用软件版本依赖不兼容问题。
4) 用户集群环境部署优化。
5) 不同虚拟集群资源逻辑隔离。

同时，本系统的设计与实现得益于计算机技术的快速发展，除了 Fat-Tree 和虚拟集群，和本文密切相关的技术还有多租户、轻量级虚拟化和高性能计算集群。

2.1 多租户

多租户是指一个单独的软件实例可以为多个组织服务。一个支持多租户的软件需要在设计上能对它的数据和配置信息进行虚拟分区，从而使每个使用这个软件的组织能使用到一个单独的虚拟实例，并且可以对这个虚拟实例进行定制化。但是要让一个软件支持多租户并非易事，因为不仅需要对它的软件架构进行相应的修改，而且对它的数据库结构也要进行特殊的设计，同时在安全和隔离性方面都要有所保障。

2.2 轻量级虚拟化

轻量级虚拟化通常称为 Linux 容器，是基于 Linux container（LXC）发展而来，目前，主要指 docker[5]。docker 是开源的轻量 OS 级虚拟化技术，主要结合了 LXC 和 UnionFS 文件系统，在近期最新的 1.9.0 版本支持了 openvswitch，生态体系逐步完善。为了应对烦琐复杂的系统环境配置与管理，OS 级轻量级虚拟化应运而生，它主要采用 cgroup、namespace 和 chroot 技术实现的，基于统一的 Linux 内核实现 OS 级应用软件封装，从而简化系统级配置与管理。

2.3 高性能计算集群

高性能计算集群通过高速物理网络互连设备，将多个节点组成的集群，并通过软件系统与高速信息交换接口，必然 MPI，组成集群系统。目前高性能计算集群主要采用 Linux 操作系统，一种典型的高性能计算集群有并行文件系统、MPI 和 SLURM[6] 组成。通常计算节点之间的带宽是影响并行计算的主要特性。物理互联设备通常采用 infiniband 或更先进的网络设备，节点间互联速度越快，集群性能就会越好。

3 Fat-Tree 与虚拟集群

Fat-Tree 网络结构是超算中心中广泛采用的一种网络架构[7]，逻辑上它是一种网络拓扑。在一种 4-ary 的 4 交换端口的三层拓扑结构，可分为边界接入、聚合和核心层。一个 k pods 的 Fat-Tree，每个包含 $\frac{k}{2}$ 交换设备的两层机构（分别是边和汇聚）。一个 k-port 交换机挂接 $\frac{k}{2}$ 个物理节点，另外 $\frac{k}{2}$ 个端口用于连接汇聚交换机，$\left(\frac{k}{2}\right)^2$ 个 k-port 核心交换机，每个核心交换机的每一个 port 用于连接每个 k-pod。每个核心交换机的第 i 个端口用于连接 pod i，每个 pod 交换机的汇聚层的连续端口连接 $\frac{k}{2}$ 跨度的核心交换机。通常情况下，k-port 交换机组成的 Fat-Tree 可以支持 $\frac{k^3}{4}$ 个节点。

基于 Fat-Tree 拓扑的虚拟集群模型有其自身的特点。首先，Fat-Tree 是可配置无阻塞的(no-blocking)，意味着它可以满足任意类型的虚拟集群请求，并且若是分配合理，可以满足用户带宽限制，提高系统的可靠性；其次，对 Fat-Tree 拓扑结构下的集群，如果满足了机架(rack)的需求(包括主机和网络带宽)，那么就能满足 rack 上虚拟集群资源分配的限制，针对这样的特点，可简化系统设计与实现。本文将虚拟集群植入分为 2 个步骤：1) rack 上资源受限的虚拟机放置；2) 受源与目的限制的虚拟链路映射。当然，这并不意味着虚拟链路映射容易或总是成功的，实际上虚拟机放置不当会对整个虚拟集群造成很不利的影响，在以后的实现中会对此做充分的论述。为了说明虚拟集群系统设计与实现问题，并明确虚拟集群构建方法的限制及求解目

标，本文对基于 Fat-Tree 的虚拟集群构建方法进行建模分析。

3.1　虚拟集群建模

物理集群建模为带权重的无向图 G_s=(N_s, L_s, A_n, A_l)，N_s 和 L_s 分别为表示物理节点和链路的集合，A_n 和 A_l 为物理节点和物理链路的属性，比如用 $A_n(c)$ 表示物理节点 CPU 的计算能力，$A_l(b)$表示物理链路的带宽。

虚拟集群请求相对物理集群，它是一种逻辑概念，为便于区分，本文将虚拟网络请求用图的顶点和边表示，建模为带权重的无向图 G_v=(V_v, E_v, R_v, R_e)，V_v 和 E_v 分别为虚拟节点(顶点)和链路(边)的集合，R_v 和 R_e 为虚拟节点和虚拟链路对物理节点和链路的请求，比如用 $R_v(c)$表示虚拟节点 CPU 的计算能力，$R_e(b)$表示虚拟链路的带宽。

虚拟集群植入通常定义为满足 G_v 到 G_s 子图的映射，并且满足 G_v 请求的 R_v 和 R_e 资源限制，M:G_v →(N', P', R_n, R_l), N'包含于 N_s，P'包含于 P_s，其中，P_s 是由 L_s 子集构成的物理路径集合。

映射可以分解为 2 个组件。

节点映射：f_n：V_v→N_s，映射虚拟节点到物理节点，并且满足对任意的 v_v 属于 V_v，存在 n_s 属于 N_s，n_s=f_n(v_v), $An(c)$≥$Rv(c)$。

链路映射：f_l：E_v→P_s，映射虚拟链路到物理路径，并且满足对任意的 e_v 属于 E_v，存在 p_s 属于 P_s，p_s=f_l(e_v), $A_n(b)$≥$R_e(b)$，其中，P_s 是由 L_s 子集构成的物理路径集合，p_s 是由 L_s 子集构成的物理路径，概念解释如表 1 所示。

表 1　虚拟集群植入问题描述

符号	解释
G_s	物理集群
N_s	物理集群节点
L_s	物理集群链路
A_n	物理集群节点属性
A_l	物理集群链路属性
P_s	物理集群路径
G_v	虚拟集群
V_v	虚拟集群节点
E_v	虚拟集群链路
R_v	虚拟机器节点资源需求
R_e	虚拟机器链路资源需求
f_n	虚拟节点到物理节点的映射函数
f_l	虚拟链路到物理路径的映射函数

图 1 展示了 2 个虚拟集群请求的植入方法。比如，虚拟集群的虚拟节点 a、b、c 和 d 分别映射到基础集群的 C、B、H 和 F，虚拟链路(a, b)、(a, c)、(a, d)、(b, d)和(c, d)分别映射到物理路径(C, A, B)、(C, G, H)、(C, D, E, F)、(B, F)和(H, F)，并且满足虚拟集群请求的所有 CPU 和带宽需求。虚拟集群请求 2 的映射也是类似的过程。

3.2　限制及目标

虚拟集群植入需要在保障 SLA 的基础上尽可能多地映射虚拟集群请求到物理集群，本文的求解目标是在最大化映射成功率并最小化资源消耗的基础上，最小化虚拟集群植入时间。借鉴相关研究

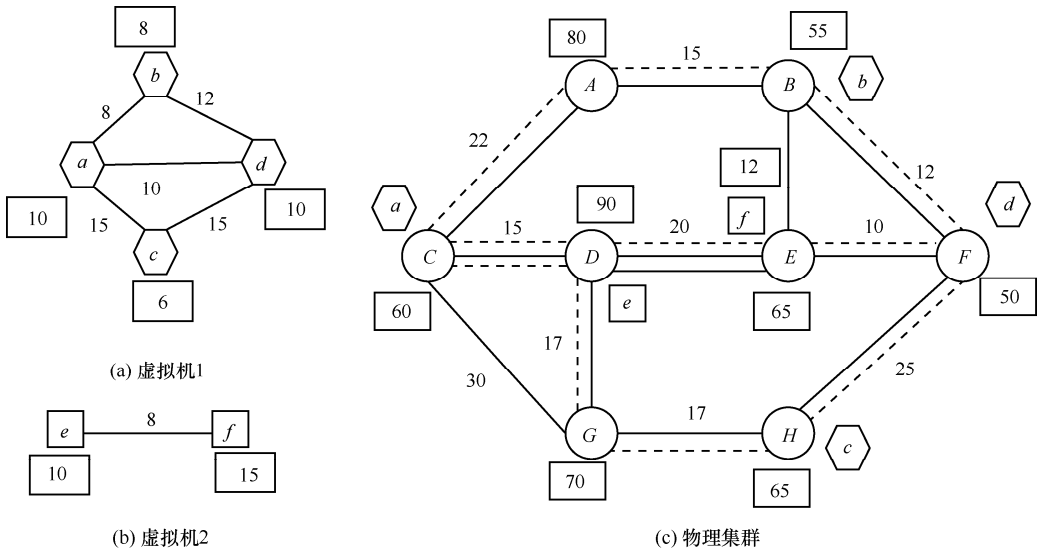

图 1　虚拟集群请求植入示例

工作[10~13]，类似于文献[9]，本文定义成功接受一个虚拟集群请求的收益为

$$R(G_v(t)) = \sum_{e_v \in Ev} BW(e_v) + \sum_{v_v \in Vv} CPU(v_v) \quad (1)$$

$BW(e_v)$ 和 $CPU(v_v)$ 分别表示 v_v 与 e_v 的带宽和 CPU 需求。

$$\mathrm{sum}T = \sum_{e_v \in Ev} T(e_v) + \sum_{v_v \in Vv} T(v_v) \quad (2)$$

$T(e_v)$和 $T(v_v)$分别表示植入单个虚拟链路和虚拟节点所消耗的时间。

$$H(v_v) = CPU(v_v) \sum_{e_v \in Ev} bw(e_v) \quad (3)$$

$bw(e_v)$和 $CPU(v_v)$ 分别表示虚拟集群请求的带宽和 CPU。

因此对每个虚拟集群请求 G_v，使在满足限制的情况下最大化 $Gv \rightarrow G's$ 成功率与收益并最小化 sum$T$ 来提高虚拟集群服务质量和用户体验的目标。结合本文系统设计所采用的 Fat-Tree 架构，并基于 openstack/scheduler[8]的方法，优化实现基于 Fat-Tree 网络的 2 阶段的虚拟集群植入方法。

4 系统架构设计

本系统设计基于 TH-Express 2+[7]三层 Fat-Tree 拓扑网络结构，虚拟 HPC 集群基础网络结构如图 2 所示的三层网络。系统中三级互连直通机柜，每个机柜有 4 个刀框，每个刀框有 32 刀片，组成密集型高性能计算集群。在第一层中，以 32 个计算节点作为计算框，接入一块称为 NRM 的交换板上，形成一级子树，NRM 板上行方向通过 20 个端口连接叶交换机。在第二层中，叶交换机由一个 24 口交换板构成，其中，下行 12 个光模块接口接入 12 个计算框，上行 12 个光模块接口接入根交换机。从叶交换机角度来看，上行链路和下行链路带宽相等，构成带宽均衡层次结构。在第三层中，根交换机是由 2 个 24 口交换板构成的 48 端口交换机，每个根交换机连接到 48 个不同的叶交换机上。页交换机和根交换机通过一个独立的路由机柜连接在一起。

每个容器部署在单个节点上，节点之间通过定制化的网桥或 openvswitch 进行网络互连通信，容器镜像为定制化的高性能软件栈，包含 MPI、Slurm 和 Lustre。基于 TH-Express 2+构建支持 openvswitch 的虚拟网络并使 docker 通过 openvswitch 进行网络通信，跨节点 docker 容器通信结构图见图 3。

本系统设计基于 openstack/nova-docker，结合高性能计算集群自身特性，系统设计同时支持物理高性能集群与虚拟 HPC 集群方式的多集群逻辑高性能软件管理模式。此种方式实现节点管理，可支持用户现有应用软件环境快速迁移、计算资源整

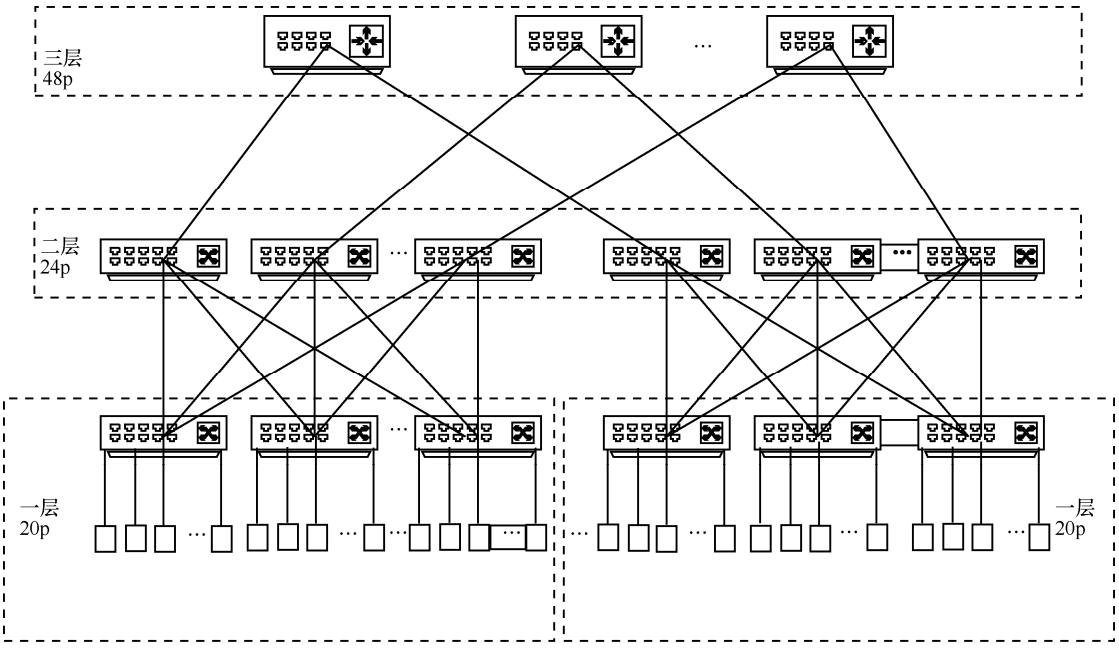

图 2 系统网络互连结构

合、用户集群环境优化部署、解决应用软件版本依赖及不兼容问题，并可实现不同集群网络逻辑隔离，同时具备云的按需付费与动态伸缩功能，从而实现一种高效的交付模式。

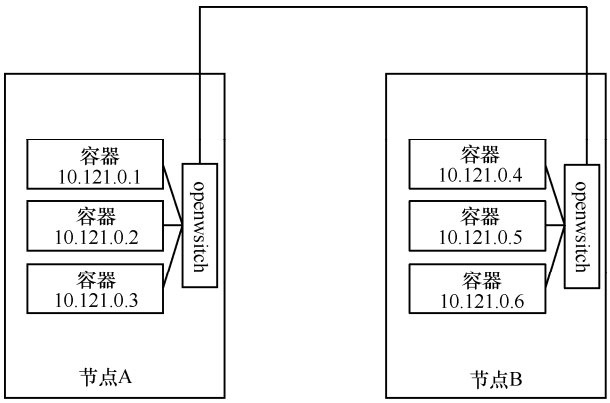

图 3　跨节点 docker 容器通信结构

另一方面，在不影响全局资源计量与计费的前提下，实现了对超算资源的优化利用。除了现有的物理 slurm 大集群，可根据用户的不同业务需求，基于高速网虚拟化技术，自建逻辑层网络，实现每个用户网络隔离的集群环境，具备单独的 slurm master 控制服务。并通过数据库记录的方式，定时将各个容器集群的资源使用情况导入原有的物理 slurm 大集群数据库中或在用户申请创建虚机 HPC 集群时，通过弱一致性方式查询、更新物理集群数据库，实现全局资源统计与计费，其系统逻辑结构见图 4。

为了构建本系统架构模型，仍需要解决如下技术问题。

4.1　基于 TH-Express 2+构建虚拟化 TCP/IP 层网络

TH-Express 2+能在物理节点或虚拟节点之间提供高带宽的基于 socket 的通信机制。在 TH-Express 2+网络体系结构中，底层为基于 RDMA 的物理网络结构。在物理层之上承载了多个层叠的虚拟网络，不同的层叠虚拟网络可用于不同的目的，如用于虚拟节点的资源管理和控制、虚拟节点间的业务通信等。最上层，通过虚拟化技术，层叠

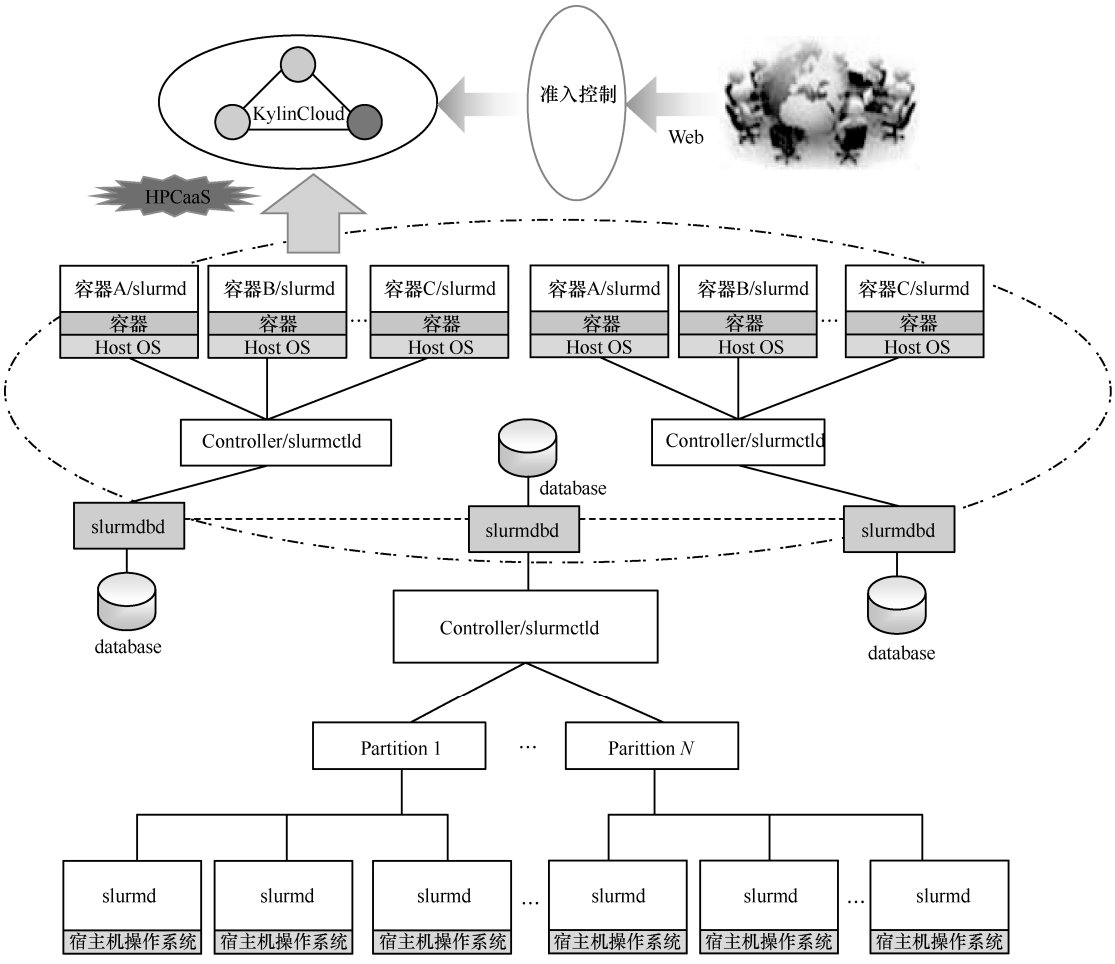

图 4　集群系统逻辑结构

的虚拟网络承载了大量的虚拟机，大数据应用和企业应用运行于虚拟机之内。基于 RDMA 的层叠虚拟网络可以让单个物理节点获得高达 42 Gbit/s 的 TCP 实测带宽，而单个虚拟节点可获得高达 10 Gbit/s 的 TCP 实测带宽，也就是说，即使是虚拟节点间的通信也获得了吉比特网络的通信能力。在第二层互连网络中，可管理的节点规模超过了 8000 多个，将很好地满足了云计算大二层的网络要求，更有利于资源的弹性分配和调度。虽然 TH-Express 2+在设计上支持虚拟化，但仍需要结构 openstack/neutron 组件结合，实现多租户网络的支持。通过移植 openvswitch/openflow 到 TH-Express 2+上，实现了对多租户虚拟网络植入与控制，并基于上文的虚拟集群模型构建虚拟 HPC 集群。

4.2 虚拟 HPC 集群植入方法

为了最小化虚拟 HPC 集群植入时间，并尽可能获得较大收益，本文采用随机搜索方法来进行虚拟机的放置问题求解，如算法 1 所示。

算法 1 虚拟集群节点植入算法

输入 G_v

输出 虚拟节点植入是否成功

步骤 1 根据收益式(1)对虚拟集群请求 $G_v=(V_v, E_v, R_v, R_e)$ 进行排序。

步骤 2 若有虚拟集群请求未被映射，则选取最大收益的请求。

步骤 3 查找满足节点约束条件 $A_n(c)\geqslant R_v(c)$的物理集群的子集 G_s'，如果不满足，则将该请求放入等待度列中。

步骤 4 植入每个虚拟节点到物理节点, 并跳转到步骤 2。

虚拟节点映射采用基于收益最大化的快速搜索方法，以减少虚拟节点映射时间开销。类似于文献[14~16],本文对可用资源的度量不简单地采用 CPU(n_s)，而是考虑 CPU 和 bandwidth 2 个参数作为可用资源的度量，如式(3)，从而有利于 VC 整体植入的成功率。

本文通过 VLAN 和 openflow 进行链路流量控制，并标示打上 tap 的网络流为逻辑的虚拟网络，根据 Fat-Tree 和 openflow 的网络跳数进行设计算法 2。

算法 2 虚拟集群链路植入算法

输入 节点成功植入的虚拟集群请求 G_v

输出 虚拟链路是否植入成功

步骤 1 对节点已成功植入的虚拟集群请求进行排序。

步骤 2 如果没有虚拟请求，则终止。

步骤 3 选择最大收益的虚拟集群请求。

步骤 4 对请求的每条虚拟链路，采用 Floyd 算法搜索 k-shortest 路径，如果找到满足带宽需求的路径，则停止搜索并将虚拟链路植入到物理路径。

步骤 5 如果步骤 4 未找到满足虚拟链路请求的路径，则推迟该请求植入并将其存入到请求队列中。

步骤 6 跳转到步骤 2。

完成虚拟集群请求的节点映射后，需要映射虚拟链路到物理路径，寻找一条虚拟链路到单个物理路径可以简化为不可分割的流问题，当其仍是 NP-Hard[17~19]。本文结合 Fat-Tree 网络拓扑的网络跳数比例，设计选择最短路径植入方法，可以快速有效地虚拟链路的映射。

4.3 服务配置与发现

为了构建虚拟 HPC 集群，仍需要解决快速镜像分发和组网。镜像分发采用 lustre 并行文件系统，将 lustre 文件系统挂载到每个节点上，通过修改 openstack 虚拟机的启动方式，实现镜像的快速分发。虚拟 HPC 集群的构建基于 openstack/Sahara 组件，通过修改部署模板来支持实现虚拟网络与虚拟 HPC 高性能集群的部署。服务发现采用 openstack 云操作系统本身的管理功能。

4.4 集群状态一致性更新

为了实现不破坏原有高性能集群管理系统又可以实现基于云的虚拟高性能集群，如文献[20]所述，本文在 openstack 的 mysql 中创建新的数据库 vhpc，来设计不同数据库之间的弱一致性更新，执行流程如下，详情如算法 3 所示。

1) 若有用户需要物理资源创建虚拟集群系统，openstack 系统发起查询功能。

2) 查询原高性能集群管理系统为被分配节点，并将该系统标记为 openstack 系统所有。

3) openstack 将查询到的节点加入 openstack 集群，并自动部署服务。

4) 云租户使用分配的资源创建虚拟高性能集群。

5) 若租户释放资源，则首先更新 openstack 的 vhpc 数据库，并调用高性能集群管理系统接口，更新该节点为为分配状态。

6) 若出现异常，则需和高性能集群管理系统数据库中节点状态保持一致。

算法 3 弱一致性更新算法

步骤 1 如果租户请求创建虚拟集，openstack

初始化查询功能。

步骤 2　从原 HPC 管理系统中查询未被分配的节点，并将该节点标记为 openstack 所有。

步骤 3　Openstack 将该节点加入到 openstack 集群并自动化部署相关服务。

步骤 4　云租户使用分配的资源创建虚拟 HPC 集群。

步骤 5　如果租户释放资源，vhpc 数据首先被更新并向 HPC 管理系统发送消息更新释放资源的节点状态。

步骤 6　如果发生异常，则保持节点状态与 HPC 管理系统数据中的状态相一致。

通过这样的异步弱一致数据库更新策略，实现两套管理系统的粗粒度资源分配与释放，从而达到在不改变原有超大规模高性能集群管理系统的基础上，实现云化的虚拟 HPC 集群。

5　系统实现

本小节描述在 KylinCloud 云平台上实现轻量级虚拟 HPC 集群的具体细节，在具体实现操作上可以分为 2 个部分：基于 KylinCloud 的虚拟 HPC 集群实现和 docker 高性能镜像制作，并借鉴文献[19,20]等相关工作，实现了一种轻量级的虚拟 HPC 云系统。

KylinCloud 是基于 openstack 开源云操作系统开发并优化的商用混合云解决方案，并且所有应用接口和官方 openstack 完全兼容。在本次系统实现上涉及到 openstack 的 nova、neutron 和 sahara 组件以及 openstack 的数据库高可用集群。

nova-docker 的一种实现是将 docker 作为一种新的 Hypervisor 来处理，作为一个 nova-compute 的 driver 集成为统一的虚拟化后端，其组件结构如图 5 所示。

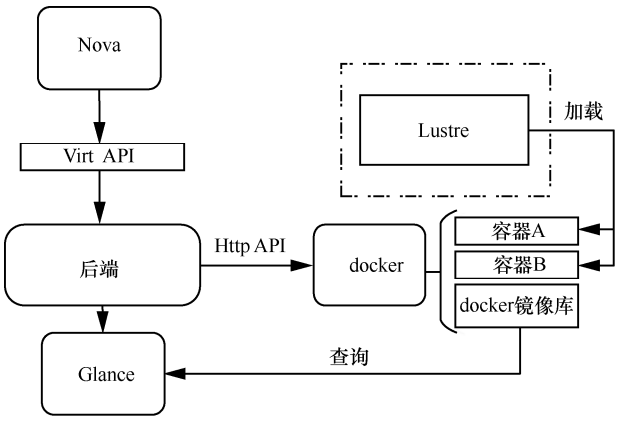

图 5　nova-docker 组件优化结构

其优点主要有以下几个方面。

1) 通过优化的 nova-scheduler 来进行资源统一调度。

2) 可用 Heat 来管理部署运行，服务发现和弹性，所有的 dockercontainer 作为 VM 来处理。

3) 通过 Neutron 来管理网络，支持 GRE、Vlan、VxLan 等，实现网络隔离。

4) 支持多租户，为不同租户设置不同的 quota。

经过实验测试，这样的一种常规的虚拟机启动逻辑在虚拟 HPC 集群部署的镜像分发上浪费了大量的时间。因为每个节点都要从 glance 下载镜像到本地，而后才能启动 docker 容器镜像。而在大量的容器镜像下载中，形成了一种瓶颈。因此本文考虑到为充分利用 lustre 并行文件系统的高并发特性，对这样的一种虚拟机启动流程进行优化。将启动镜像分为查询与加载 2 个步骤，具体步骤如下。

1) 首先查询 glance 镜像和 lustre 存储的镜像 ID 是否一致；

2) 若一致，则从 luster 直接加载镜像；若不一致，仍从 glance 中加载镜像。

通过这样的一种优化策略，极大地优化了虚拟 HPC 集群启动的时间，进而改善了用户体验，优化后的 nova-docker 组件结构见图 5。

sahara 组件在 openstack 生态中主要是用来搭建 hadoop 集群，利用虚拟出来的计算资源，快速搭建 hadoop 集群。sahara 有两种基本使用流程。

1) 快速部署，用户可以增加减少 hadoop 节点。

2) 分析即服务，根据用户设定的模板创建 hadoop 集群。

sahara 架构包含几个组件。

cluster configuration manager——集群配置管理器，这里所有的业务逻辑驻留。

auth component——身份验证组件，负责客户端身份验证和授权；

data access layer——数据访问层，持续在数据库内部模型；

VM provisioning——负责与 nova、glance 组件通信。

deployment engine——hadoop 部署引擎，可插入机制，负责部署配 VM。

所有服务通过 REST API 进行通信，而且 sahara 提供 python sahara client 功能，其类似于其他

openstack 组件中自己的 python client，在 horizon 上有 GUI，sahara 组件结构见图 6。

虚拟 HPC 集群模式类似 hadoop 的模式，本文通过第二种方式，即分析即服务。通过修改相关的少量代码和配置模板来实现虚拟 HPC 集群的集群化快速部分。数据库一致性更新通过直接调度高性能集群管理系统 mysql 数据库的 API 来实现，其具体流程如算法 3 所示。

制作高性能 docker 镜像过程比较简单，主要考虑和现有高性能集群软件兼容，故采用选择某 HPC 节点，进行文件打包，并选择节点进行测试。其制作步骤如下：

a) tar --numeric-owner --exclude=/proc–exclude =/sys –cvfhpc-base.tar /

b) mkdir hpc-base && tar -xvf hpc-base.tar -C hpc-base && cdhpc-base && tar -c . | docker import - hpc-base 或者 cat hpc-base.tar | docker import - hpc-base

c) 查看镜像，也可以直接进入 hpc-base 查看

docker images 查看生成的所有生成的镜像

docker run -i -t hpc-base/bin/bash 进终端(没有 ssh 服务)，-i 分配终端，-t 表示在前台执行，-d 表示在后台执行。

d) 根据基础镜像制作 ssh 的 docker 镜像。

e) 对制作好的镜像进行裁剪。

另外，在云系统中，为工程分配指定的资源配额，不同租户可以共享同一工程，同样的租户也可以拥有不同工程，灵活实现资源的共享与隔离。租户记录服务维护租户 ID 和 docker 之间的一个 mapping，每个 docker 作为服务的基本单元运行在实例中，从而不同租户之间可共享相同的实例或单个租户的应用运行在不同的实例中，进而可进一步优化资源利用率。

6 评估

本节从性能损耗和部署效率来比较虚拟 HPC 集群性能。硬件系统采用天河二号主机系统，系统的节点硬件环境如表 2 所示。

表 2 系统硬件配置

硬件设备	天河二号计算节点
CPU	Intel Xeon E5-2692 2.20 GHz X 2
Memory	64 GB
存储盘阵	12.4 PB
网络设备	THNI42 Gps

注：硬件环境为广州超算 TH-2 计算节点，双路 12 核 CPU，64 GB 内存，THNI（此次测试没有使用 MIC 加速卡）。

在实际部署中采用 KylinCloud V4.0 并实现高可用控制，节点为 cn807 和 cn808，用作虚拟机启动调度、网络管理、docker 镜像管理、数据库以及消息队列服务；前期部署 2 个计算节点为 cn843、cn844 用于开发调试，上面部署 docker 容器，运行时每个节点创建一个与物理节点配置相同的 docker 容器，并使用 Host 模式访问 THNI，同时选择 2 种物理节点用于分别对比 linpack 性能和 TH-Express

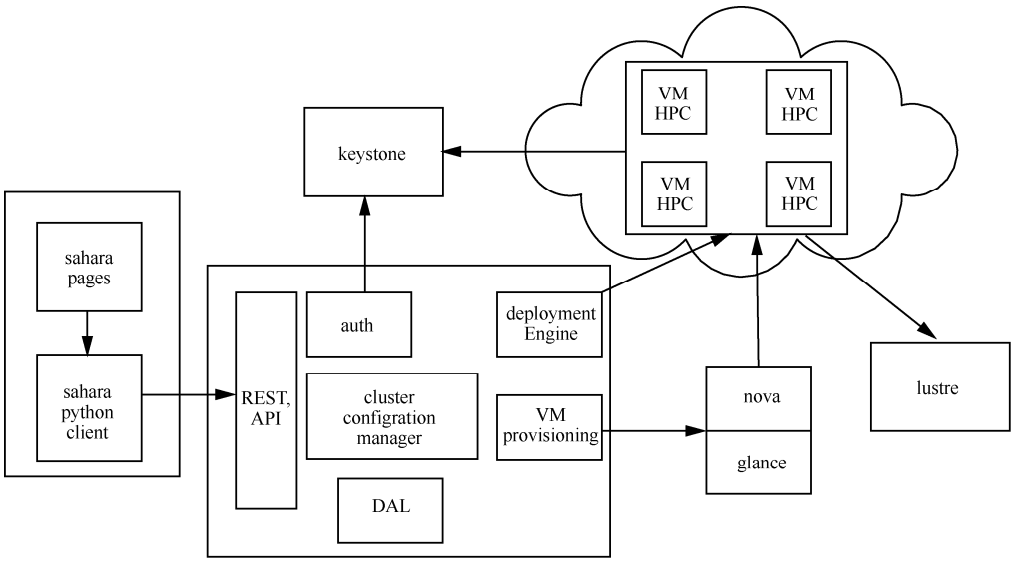

图 6 sahara 组件结构

2+（THNI）性能。

1) 用于测试虚拟 HPC 集群的节点为 cn843、cn844。

2) 用于对比测试 Linpack 性能的物理节点为 cn9331、cn9334。

3) 用于对比测试 THNI 性能的物理节点为 cn9761 和 cn9762。

Ubuntukylin14.04 用于底层物理机管理，其上运行 KylinCloud V4.0 云操作系统，软件系统版本如表 3 所示。

表 3　　软件系统

OS	Ubuntukylin x64
Docker Engine	Docker version1.90
KylinCloud	V4.0（openstack Juno）
MPI	xxx
Luster	xxx
Slurm	xxx

注：Lustre、THNI 驱动、MPI、Slurm 以及 MIC 驱动等与原 HPC 软件栈一致，高性能软件为优化后版本。

测试程序采用 Linpack 测试计算性能，版本为 HPLinpack 2.1，October 26, 2012；THNI 带宽采用 Intel (R) MPI Benchmark Suite V3.2 测试带宽损耗。测试数据如表 3 所示，经过对比可见，采用轻量级虚拟化 docker 仅产生了很少的性能损耗，分别为：在 docker 模式下相对于物理机的性能损耗为 3.32%（以时间度量）和 docker 模式下相对于物理机的性能损耗为带宽损耗为 1.18%。

表 4　　节点环境计算性能对比测试

平台/测试工具	Linpack/s	网络带宽(MB/s)
裸机	359.70	6 602.34
虚拟 HPC 集群	371.66	6 524.16
性能损耗	3.32%	1.18%

同时，为了细化说明基于 docker 的轻量级 HPC 的性能优势，本文对性能做了进一步分析。通过对比裸机、KVM 和 docker 3 种环境下的测试用例的性能值。图 7 展示了 docker 和裸机有近似的 linpack 性能值，而由于 KVM 虚拟化技术本身对 CPU 的虚拟化机制，使 vCPU 相对 CPU 有一定的性能损耗，对这种高性能应用而言，因高负荷运行且无错峰运行，分时共享 CPU 的优势无法体现，其性能损耗高可达 20%~30%。同时在网络性能方面，docker 共享物理节点的网络设备，从而性能和裸机性能相当，借助 virtio 使 KVM 的网络虚拟性能也近似裸机的性能，如图 8 所示。

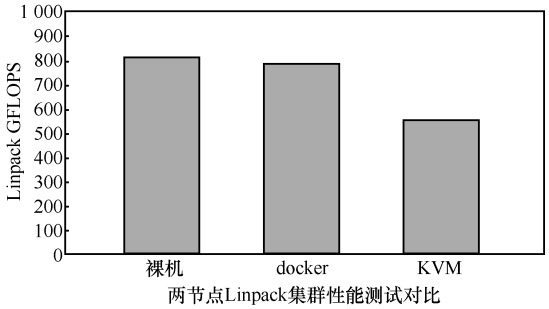

图 7　Linpack 性能比较

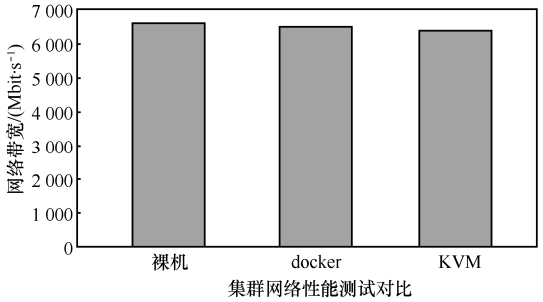

图 8　网络性能比较

另一方面，本文对云环境下虚拟 HPC 集群植入开销进行测试，通过测试分析创建不同规模的集群的创建时间在超过 20 个虚拟节点的虚拟 HPC 集群表现线性增长的趋势，主要的原因是由于创建虚拟 HPC 引发的消息队列引起的，而没有产生因下载 docker 镜像产生 IO 瓶颈，部署不同规模的虚拟 HPC 集群的时间开销如图 9 所示。

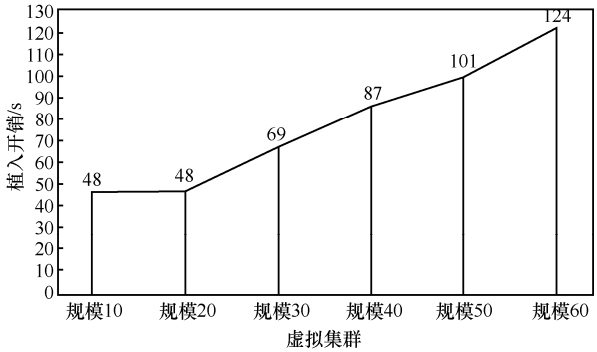

图 9　不同规模虚拟 HPC 集群部署时间测试

经过上述测试分析，可见本文的系统设计以较小的系统性能开销获得了云服务模式的按需使用的交付模式，而且在不改变现有超大规模高性能集群管理系统的前提下，可以获得灵活的使用模式和易用的应用软件部署方式，并有效地解决了原有系统可能存在的与用户应用的兼容性问题。

7 结束语

本文在不改变原有高性能集群管理系统的前提下，设计并实现了云环境下基于 Fat-Tree 和虚拟集群模型的轻量级虚拟 HPC 集群，并以很低的系统性能损耗获得了易用灵活的高性能应用服务模式。在某种程度上解决了高性能计算中心存在的软件栈僵化和应用系统部署依赖问题，交付模式问题，从而为 HPC 租户提供一种有效的交付模式。

接下来的工作主要围绕优化设计多种类型的虚拟集群植入和构建软件仓库，为不同的高性能计算应用需求的用户提供服务，比如 Java 环境等。并在现有系统的基础上实现虚拟 HPC 集群的自动伸缩服务，从而进一步有效地提高资源利用率。同时对 HPC 这类数据密集型应用，需要将并行存储上的数据传给计算节点处理，会产生大量的网络开销，采取什么样的措施优化网络传输、网络 QoS 保障以及虚拟集群的可存活性问题，将是笔者下一步研究工作的重点。

参考文献：

[1] ABDELBAKY M, PARASHAR M, JORDAN K E, et al. Enabling high-performance computing as a service[J]. IEEE Computer, 2012, 45(10): 72-80.

[2] NIU S, ZHAI J, MA X, et al. Cost-effective cloud HPC resource provisioning by building semi-elastic virtual clusters[C]//Proceedings of SC13, Networking, Storage and Analysis. 2013.

[3] XAVIER M, NEVES M, ROSSI, et al. Performance evaluation of container-based virtualization for high performance computing environments[C]//PDP,21st Euromicro International Conference on, 2013: 233-240.

[4] SLOMINSKI A, MUTHUSAMY V, KHALAF R. Building a multi-tenant cloud service from legacy code with docker containers[C]// Proc IEEE IC2E, 2015: 394-396.

[5] MERKEL D. Docker: lightweight linux containers for consistent development and deployment[J]. Linux Journal, 2014, 2014(239): 2.

[6] YOO A B, JETTE M A, GRONDONA M. Slurm: simple Linux utility for resource management[C]//Workshop on Job Scheduling Strategies for Parallel Processing. Springer, Berlin, Heidelberg, 2003: 44-60.

[7] PANG Z, WANG K, XIE M, et al. The TH express-2 high performance interconnect networks[J]. Frontiers of Computer Science, 2014, 8(3): 357-366.

[8] SEFRAOUI O, AISSAOUI M, ELEULDJ M. Openstack: toward an open-source solution for cloud computing[J]. International Journal of Computer Applications, 2012, 55(3).

[9] YU M, YI Y, REXFORD J, et al. Rethinking virtual network embedding: substrate support for path splitting and migration[J]. ACM SIGCOMM Comput. Commun. Rev., 2008, 38(2): 17-29.

[10] TANTAWI A N. A scalable algorithm for placement of virtual clusters in large data centers[C]//Proc. Int'l Symp. IEEE 20th MASCOTS, 2012: 3-10.

[11] WEI X, LI H. Topology-aware partial virtual cluster mapping algorithm on shared distributed infrastructures[J]. IEEE Transactions on Parallel and Distributed Systems, 2013: 2721-2730.

[12] LI X, WANG H, DING B, et al. Resource allocation with multi-factor node ranking in data center networks[J]. Future Generation Computer Systems, 2014, 32: 1-12.

[13] GONG S Q, CHEN J, KANG Q Y, et al. An efficient and coordinated mapping algorithm in virtualized SDN networks[J]. Front Inform Technol Electron Eng, 2016, 17(7): 701-716.

[14] PAPAGIANNI C, ANDROULIDAKIS G, PAPAVASSILIOU S. Virtual topology mapping in SDN-enabled clouds[C]//NCCA. 2014

[15] WANG Z, HAN Y, LIN T, et al. Virtual network embedding by exploiting topological information[C]//IEEE GLOBECOM, 2012.

[16] LIAO J, FENG M, LI T, et al. Topology-aware virtual network embedding using multiple characteristics[J]. KSII Trans Internet Inf Syst (TIIS), 2014, 8(1): 145-164.

[17] CHENG X, SU S, ZHANG Z, et al. Virtual network embedding through topology-aware node ranking[C]//SIGCOMM Comput Commun. 2011: 38-47.

[18] LI R Z, WU Q B, TAN Y S, et al. A cost-effective approach of building multi-tenant oriented lightweight virtual HPC cluster[C]//Cloud Computing and Big Data, CCBD, 2016.

[19] 孙大为, 常桂然, 李凤云, 等.一种基于免疫克隆的偏好多维 QoS 云资源调度优化算法[J]. 电子学报, 2011, 39(8): 1824-1831.
SUN D W, CHANG G R, LI F Y, et al. Optimizing multi-dimensional QoS cloud resource scheduling by immune clonal with preference[J]. Acta Electronica Sinica, 2011, 39(8): 1824-1831.

[20] 张尧学, 周悦芝. 一种云计算操作系统 TransOS:基于透明计算的设计与实现[J].电子学报,2011,39(5): 985-990.
ZHANG Y X, ZHANG Y X, ZHOU Y Z. A new cloud operating system: design and implementation based on transparent computing[J]. Acta Electronica Sinica. 2011,39(8):1824-1831.

[21] 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 X, ZHANG Z B, SU S, et al. Survey of virtual network embedding problem[J]. Journal on Communications, 2011, 32(10): 143-151.

[22] 江逸茗, 兰巨龙, 程东年, 等. 分布式环境中基于协商的虚拟网映射算法[J]. 通信学报, 2014, 35(12): 62-69.
JIAN Y M, LAN J L, CHENG D N, et al. Virtual network embedding algorithm based on negotiation in distributed enlivenment[J]. Journal on Communications, 2014, 35(12): 62-69.

作者简介：

谭郁松（1976-），男，江西余干人，博士，国防科技大学研究员，主要研究方向为操作系统、云计算与大数据。

李荣振（1985-），男，河南周口人，博士，国防科技大学博士生，主要研究方向为云计算与网络虚拟化。

吴庆波（1969-），男，浙江宁波人，博士，国防科技大学研究员，主要研究方向为操作系统与云计算。

张建锋（1984-），男，陕西宝鸡人，博士，国防科技大学助理研究员，主要研究方向为操作系统、网络安全与云计算。

张尧学（1956-），男，湖南常德人，博士，中国工程院院士，中南大学教授、博士生导师，主要研究方向为计算机网络、操作系统与透明计算。