基于流量演化感知的服务功能链在线弹性编排策略

谷允捷,胡宇翔,丁悦航,谢记超
(国家数字交换系统工程技术研究中心,河南郑州 450002)

摘 要: 随着网络功能虚拟化(NFV)的引入,运营商能够提供更为灵活的网络服务.然而现有服务功能链(SFC)编排局限于静态反应式策略,业务流量发生变化时网络资源供应量难以匹配负载需求,虚拟网络功能(VNF)频繁部署与迁移,运营开销增大.针对上述问题,该文提出一种基于流量演化感知的服务功能链在线弹性编排策略(OEOP),该策略将在线学习引入到 SFC 流量演化感知的过程,预先确定细粒度的 VNF 弹性需求.此外,以实时更新的 SFC 路径与节点负载两因子为导向,完成新增 VNF 的在线弹性部署,代替 VNF 迁移应对系统负载变化.仿真表明,该策略明显增强了虚拟资源供应量与负载需求的匹配特性,VNF 吞吐量利用率提高 10.2% ~24.8%,运营开销平均降低 26.7%.

关键词: 网络功能虚拟化;虚拟网络功能;运营开销;流量演化感知;在线凸优化
中图分类号: TP393.0 **文献标识码:** A **文章编号:** 0372-2112 (2019)10-2192-10
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2019.10.023

Online Elastic Orchestration Policy of Service Function Chain Based on Flow Evolution Perception

GU Yun-jie, HU Yu-xiang, DING Yue-hang, XIE Ji-chao
(*National Digital Switching System Engineering and Technological R&D Center, Zhengzhou, Henan* 450002, *China*)

Abstract: With the introduction of network function virtualization (NFV), operators can provide more flexible network service. However, most existing orchestration schemes of service function chain (SFC) are limited to the static or reactive policy where the virtual network functions (VNFs) need to be deployed or migrated frequently, and will easily lead to the mismatch of service supplement and high operational expenditure under time-varying workload. This paper proposes an online elastic orchestration policy (OEOP) based on the evolution perception of flow rate to solve the above mentioned problem. OEOP introduces online learning into the evolution perception of flow rate, which helps to predetermine the fine-grained VNF scaling demands. In addition, the online elastic deployment is achieved according to the real-time update information of SFC paths and the load of nodes. The newly deployed VNF instances can respond to the time-varying workload by taking place of the mission of VNF migration. The simulation results demonstrate that OEOP can significantly enhance the matching between virtual resource supply and workload demand. The throughput of VNF is improved by 10.2% ~24.8%, and the operational expenditure can be reduced by 26.7% on average compared with other solutions.

Key words: network function virtualization; virtual network functions; operational expenditure; flow evolution perception; online convex optimization

1 引言

当前多样化的互联网业务发展迅猛,而僵化的网络架构与臃肿的专用网元设备难以应对快速变化的业务需求.扩建基础设施的对策导致网络规模急剧膨胀,迫使收益下滑的运营商不得不面对网络架构转型,打破专用硬件设备的垄断[1].网络功能虚拟化(NFV,Network Function Virtualization)依托云计算与虚拟化技术,

收稿日期:2018-12-04;修回日期:2019-03-13;责任编辑:蓝红杰
基金项目:国家自然科学基金(No.61521003,No.61872382);国家重点研发计划课题(No.2017YFB0803204)

使虚拟网络功能(VNF,Virtual Network Function)与专用硬件设备完全解耦,支持新兴业务的快速上线,极大地降低了运营成本. 由多个 VNF 按业务需求串联形成的虚拟路径称为服务功能链[2](SFC, Service Function Chain),SFC 为网络业务提供了高度灵活,资源节约和集中编排的特性,成为当前研究热点.

SFC 编排相关研究在 VNF 部署与资源优化策略上取得了丰富成果,也有一些文献关注了虚拟资源弹性伸缩的需求. Moens H 等[3]以最小化服务器数量为优化目标,构建 ILP 模型实现小规模网络中的精确求解. Bari F[4]综合了 VM 部署,运行消耗与转发代价等因素,在大小型网络中分别设计了快速编排策略. Jang I 等[5]将 SFC 接受率和能源消耗建模为多目标优化问题并求解. Ghaznavi M 等[6]创造性地提出一种分布式 SFC 策略,解耦了 VNF 吞吐量对服务器容量的依赖. Cohen R 等[7]以最优化 VNF 之间的传输距离为目标,提出具有严密理论分析的优化算法. Wang X 等[8]关注 VM 部署与运行能耗,对单、多条 SFC 分别了设计相应的 VNF 最小供应开销算法. Arteaga C 等[9]在 vEPC 中结合 Q-learning 与高斯回归,实现 VNF 数目随流量变化动态调整. Zhang X 等[10]提出一种 VNF 弹性伸缩方法,通过决策 VNF 租期保证 QoS 并节约资金. Fei X 等[11]假设 VNF 吞吐量可调整,借助具有自适应流量变化能力的 VNF 提高了资源利用率. Wang M 等[12]将 VM 合并建模为随机装箱问题,并设计在线装箱算法求解,降低 30% 的资源占用. Wen T 等[13]基于文献[12]提出的 VM 合并思想,设计 VNF 合并方案减少 VNF 实例,相对文献[14]平均降低了 32% 的 VNF 数目. Chen Z 等[15]针对业务流动态特性引起资源利用率下降的问题,设计 SFC 迁移重配置策略,在保证业务流服务质量的同时有效改善了资源利用率.

虽然现有的 VNF 部署方案较好地解决了 VNF 初始分配问题,但是在实际网络中,SFC 数量与流量的动态特性将引起 VNF 需求量不断变化[16,17],导致“静态”的 VNF 部署策略得出的虚拟资源初始分配方案很快无法匹配 SFC 时变流量. 而现有的弹性伸缩方案只能对已达 NFV 系统的流量做出反应式决策,算法的决策过程以及 VNF 的增删操作引起的时延可能导致 SLA(Service-Level Agreement)违例[18]. 因此,“静态反应式策略”尚未实现虚拟资源随负载需求的自适应调整,削减了 VNF 相对专用硬件设备在虚拟资源可伸缩方面的优势.

针对上述问题,本文提出一种基于流量演化感知的 SFC 在线弹性编排策略,首先借助在线学习算法感知 SFC 流量变化,预先确定细粒度的 VNF 弹性需求,减少轻载 VNF;其次设计在线弹性部署算法,在新增工作负载到达前,完成 VNF 弹性部署与流量路径规划. 实现虚拟资源供应量与业务负载变化的匹配,降低 NFV 运营商总开销.

2　数学模型与问题描述

2.1　系统模型

底层网络

代表物理资源的底层网络由 $G=(N,E,A_n,A_e)$ 表示. 其中 N 为节点集合,E 为链路集合. A_n 为 N 中节点 n 的属性,可表示为 $A_n=\{C_n^r,L_n\}$,C_n^r 表示不同类型资源总量,$r\in R=\{\mathrm{CPU},\mathrm{RAM},\mathrm{HDD}\}$ 分别表示计算、内存、存储资源,L_n 代表节点位置. $A_e=\{B_e,\delta_e\}$ 表示 E 中链路 e 的属性,B_e 为链路带宽容量,δ_e 为单位流量传输代价.

虚拟网络功能(VNF)

VNF 类型用 I 表示,属性为 $A^i=\{C_i^r,B_i,\varepsilon_i,\mu_i,\lambda_i\}$,其中 C_i^r 为 VNF 对资源 r 的需求量,B_i 为其吞吐量上限,一个 VNF 可以同时处理多条 SFC,直到流量达到吞吐量上限. ε_i 为 i 型 VNF 的部署代价,μ_i 为 VNF 单位时间的运行开销,λ_i 为 i 型 VNF 流量增衰比,代表 VNF 处理引起的流量放缩.

服务功能链(SFC)

J 为 SFC 集合,定义五元数组 $A^j=\{s_j,o_j,\beta_j,\alpha_j(t),\tau_j\}$ 为 SFC 属性. $s_j,o_j\in N$ 分别为链 j 的源节点和目的节点,β_j 为组成该 SFC 的 VNF 序列,如 Firewall→IDS→Proxy. 设 NFV 系统运行总时间为 T,$t=\{1,2,3\cdots\}$ 代表时隙为 Δt 的离散时间,$\alpha_j(t)$ 为 t 时刻 SFCj 的流量,τ_j 为其生命周期,设 SFC 于 t_0 时刻发起,则当 $t_0\leqslant t\leqslant t_0+\tau_j$ 时,$\alpha_j(t)>0$,当 $t\in[0,t_0)\cup(t_0+\tau_j,T]$ 时,$\alpha_j(t)=0$.

为描述底层网络与 SFC 编排的关系,定义变量 $x_i(t)$ 表示 t 时刻底层网络中 i 型 VNF 数量,$x_{i_k}^n(t)$ 表示 t 时刻第 k 个 i 类型 VNF 是否部署于节点 n,$x_i^n(t)$ 为节点 n 上部署 i 型 VNF 的数量,变量 $\upsilon_j^{e(u,v)}(t)$ 表示 SFC 是否经过链路 e,$\upsilon_j^{i_k}(t)$ 表示 SFC 是否由 VNFi_k 处理.

$$x_{i_k}^n(t)=\begin{cases}1, & \text{第 } k \text{ 个 } i \text{ 型 VNF 部署在节点 } n\\ 0, & \text{第 } k \text{ 个 } i \text{ 型 VNF 不部署在节点 } n\end{cases}\tag{1}$$

$$x_i^n(t)=\sum_{k\in x_i(t)}x_{i_k}^n(t)\tag{2}$$

$$\upsilon_j^{e(u,v)}(t)=\begin{cases}1, & \text{SFC } j \text{ 经过链路 } e(u,v)\\ 0, & \text{SFC } j \text{ 不经过链路 } e(u,v)\end{cases}\tag{3}$$

$$\upsilon_j^{i_k}(t)=\begin{cases}1, & \text{SFC } j \text{ 经过第 } k \text{ 个 } i \text{ 型 VNF 处理}\\ 0, & \text{SFC } j \text{ 不经过第 } k \text{ 个 } i \text{ 型 VNF 处理}\end{cases}\tag{4}$$

2.2　优化目标

为描述虚拟资源供求匹配特性与优化目标,定义以下指标.

定义 1　负载强度 $W(t)$. $W_n(t)$ 表示节点负载强度,即节点 n 的 CPU 占用率,$W_{i_k}(t)$ 代表单个 VNF 负载

强度,表示该 VNF 吞吐量利用率,$\alpha_j^{i_k}(t)$代表 SFC 经该 VNF 处理时的流量大小. $\overline{W}_N(t)$,$\overline{W}_{\text{VNF}}(t)$分别表示节点平均负载强度和 VNF 平均负载强度.

$$W_n(t) = \frac{\sum_{i\in I} x_i^n(t)\cdot C_i^{\text{CPU}}}{C_n^{\text{CPU}}}, \overline{W}_N(t) = \frac{\sum_{n\in N} W_n(t)}{N} \tag{5}$$

$$W_{i_k}(t) = \frac{\sum_{j\in J} v_j^{i_k}(t)\cdot \alpha_j^{i_k}(t)}{B_i}, \overline{W}_{\text{VNF}}(t) = \frac{\sum_{i\in I}\sum_{k\in x_i(t)} W_{i_k}(t)}{\sum_{i\in I} x_i(t)} \tag{6}$$

定义 2 VNF 运行开销 $O(t)$. VNF 运行开销与各类 VNF 数目及其运行时间 t 有关,可表示为

$$O_{\text{total}}(t) = \sum_{t\in T}\sum_{n\in N}\sum_{i\in I}\mu_i x_i^n(t) = \sum_{t\in T}\sum_{i\in I}\mu_i x_i(t) \tag{7}$$

设 t 时刻存在 VNF 数目最优值 $\tilde{x}_i(t)$能恰好处理所有 SFC 流量,则由于 VNF 供大于求导致的额外运行开销可描述为

$$O_{\text{extra}}(t) = \sum_{t\in T}\sum_{i\in I}\mu_i[x_i(t) - \tilde{x}_i(t)] \tag{8}$$

定义 3 VNF 部署开销 $D(t)$. 部署 VNF 需要传输并启动 VM 镜像,这些操作带来一定开销. 应注意到 VNF 部署总开销 $D_{\text{total}}(t)$与底层节点 n 有关,考虑到 VNF 迁移的影响,若 $t-1$ 时刻一个位于节点 n_1 的 i 型 VNF 在 t 时刻迁移到节点 n_2,虽然 i 型 VNF 总数没有变化,但 n_2 节点上增加了一次部署操作. VNF 部署总开销可以表示为式(9),其中[·]$^+$运算符定义为$[f(x)]^+ = \max\{f(x),0\}$.

$$D_{\text{total}}(t) = \sum_{t\in T}\sum_{n\in N}\sum_{i\in I}\varepsilon_i[x_i^n(t) - x_i^n(t-1)]^+ \tag{9}$$

当系统不存在 VNF 迁移的情况时,VNF 部署开销可以表示为式(10).

$$D_{\text{deploy}}(t) = \sum_{t\in T}\sum_{i\in I}\varepsilon_i[x_i(t) - x_i(t-1)]^+ \tag{10}$$

由 VNF 迁移导致的部署开销可以定义为:

$$\begin{aligned} D_{\text{migrate}}(t) &= D_{\text{total}}(t) - D_{\text{deploy}}(t) \\ &= \varepsilon_i\sum_{t\in T}\sum_{i\in I}\Big\{\sum_{n\in N}[x_i^n(t) - x_i^n(t-1)]^+ \\ &\quad - [x_i(t) - x_i(t-1)]^+\Big\} \end{aligned} \tag{11}$$

定义 4 损失服务开销 $L(t)$. VNF 供应不足会导致部分流量无法得到服务,此时 NFV 运营商必须使用私有服务器集群处理为这部分流量,可表示为式(12),ψ为服务器集群处理单位流量的价格,$\alpha_j^{\text{loss}}(t)$为 t 时刻未被已部署 VNF 接收的流量.

$$L(t) = \sum_{t\in T}\sum_{j\in J}\psi\alpha_j^{\text{loss}}(t) \tag{12}$$

定义 5 流量传输开销 $R(t)$. 在物理链路上搭建虚拟链路并传输数据会引起流量传输开销,可表示为式(13),$\alpha_j^e(t)$表示 SFC 经过链路 e 时的流量大小.

$$R(t) = \sum_{t\in T}\sum_{j\in J}\sum_{e(u,v)\in E}\delta_e v_j^{e(u,v)}\alpha_j^e(t) \tag{13}$$

基于以上定义,NFV 运营商累积开销为

$$\begin{aligned} C(t) &= O_{\text{total}}(t) + D_{\text{total}}(t) + L(t) + R(t) \\ &= \sum_{t\in T}\Big\{\sum_{i\in I}\{\mu_i x_i(t) + \sum_{n\in N}\varepsilon_i[x_i^n(t) - x_i^n(t-1)]^+\} \\ &\quad + \sum_{j\in J}\{\psi\alpha_j^{\text{loss}}(t) + \sum_{e(u,v)\in E}\delta_e v_j^{e(u,v)}\alpha_j^e(t)\}\Big\} \end{aligned} \tag{14}$$

NFV 运营商的优化目标为 min$C(t)$,同时应满足式(15)~(19)约束关系

St:

$$\sum_{i\in I} x_i^n(t)C_i^r \leqslant C_n^r, \forall n\in N, r\in R, t\in T \tag{15}$$

$$\sum_{j\in J} v_j^{e(u,v)}\alpha_j^e(t) \leqslant B_e, \forall e\in E, t\in T \tag{16}$$

$$\sum_{n\in N} x_i^n(t) = x_i(t), \forall i\in I, t\in T \tag{17}$$

$$\sum_{n\in N} x_{i_k}^n(t) = 1, \forall n\in N, t\in T, 0\leqslant k\leqslant x_i(t) \tag{18}$$

$$\sum_{v\in N} v_j^{e(w,v)}\alpha_j(t) - \sum_{u\in N} v_j^{e(u,w)}\alpha_j(t) = 0, \quad \forall t\in T, w\in N\backslash\{s_j,o_j\} \tag{19}$$

式(15)表示节点承载的 VNF 不得超出其资源上限,式(16)表示链路承载的流量不得超出其带宽容量,式(17)表示节点内 i 型 VNF $x_i^n(t)$数量与其总数量 $x_i(t)$的关系,式(18)约束了一个 VNF 只能部署在一个节点上,式(19)表示除 SFC 源、目的节点外,任何中间节点遵守流守恒约束.

3 算法设计

基于流量演化感知的在线弹性编排策略(OEOP)首先设计流量演化感知算法(FEPA,Flow Evolution Perception Algorithm)感知同构 SFC 的流量变化,进而预测下一时刻所需的 VNF 增量;然后通过基于双因子协同导向的在线弹性部署算法(BCGOD,Bivariate Co-operation Guidance Online Elastic Deployment Algorithm)结合预测增量进行 VNF 的预先部署和 SFC 路径规划,最终实现虚拟资源的供应随 NFV 工作负载变化主动调整以节约 NFV 运营商开销.

3.1 SFC 流量演化感知模型

SFC 可由用户随时发起,且短生命周期的 SFC 难以为感知流量变化提供充足的信息,所以单条 SFC 流量变化往往不具备可预测性. 为感知 SFC 流量变化,定义同构服务功能链作为研究对象:

定义 6 同构服务功能链(HSFC,Homogeneous SFC). HSFC 是指 $\forall j_1,j_2\in J$,若$\{s_{j_1},o_{j_1},\beta_{j_1}\} = \{s_{j_2},o_{j_2},\beta_{j_2}\}$,则 SFC j_1,j_2 属于一条同构服务功能链 $\bar{j}$,$\bar{j}$ 的流量为 $\alpha_{\bar{j}}(t)$,$\alpha_{\bar{j}}(t) = \sum_{j\in\bar{j}}\alpha_j(t)$,由同构服务链 $\bar{j}$ 组成的集

合为 $\bar{J}$.

流量演化感知模型对系统中 HSFC 流量 $\alpha_j(t)$ 的变化进行预测,设在 t 时刻得到的感知结果为 $\alpha_j^*(t+1)$,$\alpha_j^*(t+1)$ 与实际值 $\alpha_j(t+1)$ 之间的误差将直接影响 VNF 供应:若 $\alpha_j^*(t+1)<\alpha_j(t+1)$,会造成 VNF 供小于求,部分流量无法得到服务,增加 $L(t)$;若 $\alpha_j^*(t+1)>\alpha_j(t+1)$,则会造成 VNF 供大于求,增加 $O_{\text{extra}}(t)$.

流量数据时效性高,需要快速得出感知结果,因此流量演化感知算法应兼顾准确性与高效性. 在线学习旨在利用实时数据和历史判决值做出快速在线预测,对处理在线数据流有着先天性优势,近年来在广告点击率预估和云计算在线预测领域已经取得显著成果[19,20]. 本文基于在线学习算法 FTRL-Proximal 设计 FEPA 算法感知流量变化.

3.1.1　在线学习与流量演化感知算法(FEPA)

预测策略集 α_j^*　依据流量历史数据可知 $\alpha_j(t)$ 的最大值为 $\alpha_j^{\max}$,则定义预测策略集 $\alpha_j^*\in[0,\alpha_j^{\max}]$.

感知损失函数 $f_t[\alpha_j^*(t)]$　感知损失函数定义为 t 时刻预测流量值与真实流量值的平方差,即

$$f_t[\alpha_j^*(t)]=[\alpha_j^*(t)-\alpha_j(t)]^2 \tag{20}$$

***regret* 函数** reg_T　regret 函数衡量了系统运行时间 T 内,算法得出的累积损失与最优离线解 α_j^{static} 得出的累积损失之差,其增长特性是衡量感知效果的重要指标. regret 函数可表示为式(21)

$$\begin{aligned}&\text{reg}_T=\sum_{t=1}^{T}f_t[\alpha_j^*(t)]-\sum_{t=1}^{T}f_t[\alpha_j^{\text{static}}],\forall\bar{j}\in J\\&\alpha_j^{\text{static}}=\arg\min_{\alpha}\sum_{t=1}^{T}f_t(\alpha),\alpha\in[0,\alpha_j^{\max}]\end{aligned} \tag{21}$$

由于策略集 α_j^* 是连续凸集,$f[\alpha_j^*(t)]$ 是关于 $\alpha_j^*(t)$ 的凸函数,因此流量演化感知可视为在线凸优化问题,寻找合适的感知更新策略,实现 reg_T 随时间次线性增长即可保证感知有效,且 reg_T 值越低说明感知效果越好[20]. reg_T 次线性增长可表示为

$$\lim_{t\to\infty}\frac{\text{reg}_T}{t}=\lim_{t\to\infty}\frac{\sum_{t=1}^{\infty}\{f_t[\alpha_j^*(t)]-f_t[\alpha_j^{\text{static}}]\}}{t}=0,\forall\bar{j}\in J \tag{22}$$

为实现该目标,一个最直观的方法是利用历史数据每次找到能使累积误差最小的值作为最优解,这种在线优化方法称为 FTL(Follow The Leader),可表示为:

$$\alpha_j^*(t+1)=\underset{\alpha_j^*}{\text{argmin}}\sum_{s=1}^{t}f_s(\alpha_j^w),\alpha_j^w\in\alpha_j^*,\forall\bar{j}\in J \tag{23}$$

然而 FTL 算法对应的 reg_T 上界在 $\alpha_j^*(t)$ 上下震荡时可能达到 $O(T)$,不能保证 reg_T 函数的次线性增长[21],因此 FTL 无法保证预测的准确性. FTRL-Proximal 算法在 FTL 算法的基础上,增加了关于 $\alpha_j^*(t)$ 的凸函数正则项 $R(\alpha_j^w)$,使算法更加平稳,其感知更新策略为:

$$\alpha_j^*(t+1)=\underset{\alpha_j^*}{\text{argmin}}\left\{\sum_{s=1}^{t}f_s(\alpha_j^w)+R(\alpha_j^w)\right\},\alpha_j^w\in\alpha_j^*,\forall\bar{j}\in J \tag{24}$$

为减少计算复杂度,使用代理函数来对感知损失函数进行简化. 令 $g_{jt}=\partial f_t[\alpha_j^*(t)]$ 表示 $f_t[\alpha_j^*(t)]$ 的次梯度,则代理函数可表示为 $g_{jt}\alpha_j^*(t+1)$,可得 FEPA 感知更新策略:

$$\begin{aligned}&f_{\text{FEPA}}(\alpha_j^w)=\sum_{s=1}^{t}g_{js}\alpha_j^w+\frac{1}{2}\sum_{s=1}^{t}\sigma_s\|\alpha_j^w-\alpha_j^*(s)\|_2^2+L_2\|\alpha_j^w\|_2^2\\&\alpha_j^*(t+1)=\underset{\alpha_j^*}{\text{argmin}}\{f_{\text{FEPA}}(\alpha_j^w)\},\alpha_j^w\in\alpha_j^*,\forall\bar{j}\in J\end{aligned} \tag{25}$$

$f_{\text{FEPA}}(\alpha_j^w)$ 第一项是代理函数对感知损失函数的估计,其中 $g_{jt}=\partial f_t[\alpha_j^*(t)]=2[\alpha_j^*(t)-\alpha_j(t)]$;第二项限制了每次更新结果不得与已经迭代过的结果相差太大,避免了流量值震荡导致的 reg_T 函数上界达到 $O(T)$,其中 $\sigma_t=\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}}$ 表示学习率随时间更新策略,$\frac{1}{\eta_t}$ 是随时间变化的非增序列,FEPA 算法对 $\frac{1}{\eta_t}$ 的更新公式为 $\eta_t=\frac{\theta}{G\sqrt{t}}$,$\theta=\alpha_j^{\max}$,$\|g_{jt}\|\leqslant G$;第三项的作用是使预测结果相对平滑. FEPA 模型求解效率高,当 t 固定时 $f_{\text{FEPA}}(\alpha_j^w)$ 是关于 α_j^w 的光滑凸函数,因此可通过求凸函数极值的方式,令 $\partial f_{\text{FEPA}}(\alpha_j^w)=0$,即可得到 $\alpha_j^*(t+1)$.

3.1.2　FEPA 算法可行性分析

通过证明 FEPA 算法的 reg_T 上界的次线性来说明流量演化感知可行性[21].

引理 1　FTRL-Proximal 边界　$f_{\text{FTRL}}(\cdot)$ 可表示为 $f_{\text{FTRL}}=\sum_t f_t+\sum_t r_t$ 的形式,$g_{jt}=\partial f_t$,则其 reg_T 上界为

$$\text{reg}_T\leqslant\sum_{t=0}^{T}r_t+\frac{1}{2}\sum_{t=1}^{T}\frac{1}{\sigma_{1:t}}\|g_t\|_2^2 \tag{26}$$

引理 1 证明见文献[22].

定理 1　系统运行时间为 T 时,FEPA 算法得出的 reg_T 满足次线性增长要求,且上界为

$$\text{reg}_T\leqslant\frac{1}{2}G\alpha_j^{\max}(3\sqrt{T}-1),\forall\bar{j}\in\bar{J} \tag{27}$$

证明　FEPA 算法中,

$$r_t=\begin{cases}0, & t=0\\ \frac{\sigma_t}{2}\|\alpha_j^w-\alpha_j^*(t)\|_2^2, & t\geqslant1\end{cases},$$

由引理 1 可知

$$\text{reg}_T\leqslant r_{0:T}+\frac{1}{2}\sum_{t=1}^{T}\frac{1}{\sigma_{1:t}}\|g_t\|_2^2$$

$$=\frac{1}{2}\sum_{t=1}^{T}\left(\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}}\right)\left[\alpha_j^{w}-\alpha_j^{*}(t)\right]^2+\frac{1}{2}\sum_{t=1}^{T}\eta_t g_t^2$$

$$\leqslant\frac{1}{2}\frac{1}{\eta_T}(\alpha_j^{\max})^2+\frac{1}{2}\sum_{t=1}^{T}\eta_t G^2$$

$$=\frac{1}{2}\left[G\sqrt{T}(\alpha_j^{\max})+\alpha_j^{\max}\left(1+\int_{t=1}^{T}\frac{\mathrm{d}t}{\sqrt{t}}\right)G\right]$$

$$=\frac{G\alpha_j^{\max}(3\sqrt{T}-1)}{2}$$

可见$\lim\limits_{T\to\infty}\frac{\mathrm{reg}_T}{T}\leqslant\lim\limits_{T\to\infty}\frac{G\alpha_j^{\max}(3\sqrt{T}-1)}{2T}=0$, $\forall\bar{j}\in J$,满足次线性增长条件,流量演化感知有效.

3.1.3 VNF 增量的确定

任意时刻,NFV 系统中各类型 VNF 吞吐量之和须满足所有 HSFC 流量处理需求,可表示为式(28),其中$\lambda_i^j=\prod_{s=0}^{k-1}\lambda_s$代表累积流量增衰比,$k$ 表示 VNFi 在序列β_j中的位置,λ_s 表示位置 s 对应 VNF 的增衰比,其中 $\lambda_0=1$.

$$x_i(t)B_i\geq\sum_{j\in J}\lambda_i^j\alpha_j(t),\ \forall t\in T,i\in I \tag{28}$$

兼顾流量处理需求与 VNF 资源节约,$t+1$ 时刻所需的各种 VNF 最优数目可表示为

$$\tilde{x}_i(t+1)=\frac{\sum_{j\in J(t+1)}\prod_{s=0}^{k-1}\lambda_s\alpha_j^{*}(t+1)}{B_i}=\frac{\sum_{j\in J(t+1)}\lambda_i^j\alpha_j^{*}(t+1)}{B_i},\quad\forall t\in T \tag{29}$$

但 $\tilde{x}_i(t+1)$不是整数,因此需要对 $\tilde{x}_i(t+1)$进行取整,取整策略需满足取整后 $x_i(t+1)$的期望 $E[x_i(t+1)]=\tilde{x}_i(t+1)$,因此取整概率分布可表示为

$$F(\tilde{x}_i(t+1))=\begin{cases}P[x_i(t+1)=\lceil\tilde{x}_i(t+1)\rceil]=1+\lfloor\tilde{x}_i(t+1)\rfloor-\tilde{x}_i(t+1)\\P[x_i(t+1)=\lfloor\tilde{x}_i(t+1)\rfloor]=\tilde{x}_i(t+1)-\lfloor\tilde{x}_i(t+1)\rfloor\end{cases} \tag{30}$$

取整后的 $x_i(t+1)$即为 $t+1$ 时刻最优 VNF 数目,感知增量 $\Delta x_i(t+1)=x_i(t+1)-x_i(t)$代表 $t+1$ 时刻的 VNF 弹性伸缩需求.

3.2 基于双因子协同导向的在线弹性部署算法(BCGOD)

BCGOD 算法结合感知增量 $\Delta x_i(t+1)$,在 t 时刻预先完成 $t+1$ 时刻的 VNF 弹性伸缩需求与新增流量路径规划. BCGOD 分为相互作用的两个阶段,第一阶段以 t 时刻已构建的 SFC 路径与节点负载两因子为导向决策 VNF 弹性部署,降低 $D(t)$,建立“虚拟资源网络”;第二阶段在“虚拟资源网络”的基础上采用节点分割与动态规划思想,基于 Viterbi 算法为 $t+1$ 时刻新增 SFC 构建传输路径,降低 $R(t)$,新建路径又会影响下一时刻“虚拟资源网络”的构建.

3.2.1 虚拟资源网络的构建

为清晰描述“虚拟资源网络”构建算法,首先引入新增 SFC、节点 VNF 介数权重、节点-VNF 吸引度等概念.

定义 7 新增 SFC. 对于 $\Delta\alpha_j^{*}(t+1)=\alpha_j^{*}(t+1)-\alpha_j(t)>0$ 的 HSFC,可将预测增量 $\Delta\alpha_j^{*}(t+1)$视为 $t+1$ 时刻的新增 SFC,记为 $\Delta\bar{j}(t+1)$,NFV 系统中 $t+1$ 时刻新增 SFC 集合记为 $\Delta\bar{J}(t+1)$,代表了 $t+1$ 时刻系统新增工作负载.

定义 8 节点-VNF 介数权重(NBW,Node-VNF Betweenness Weight). 每个底层节点对所有 VNF 类型维持相应权重值,代表 t 时刻含有 i 型 VNF 的 SFC 路径 Path($\Delta\bar{j}(t)$)经过节点 n 的累计次数,计算公式为

$$\mathrm{NBW}_n^i(t)=\sum_{\Delta\bar{j}\in\Delta\bar{J}(t)}\sum_{k\in x_i(t)}\upsilon_{\Delta\bar{j}}^{i_k}\cdot x_{i_k}^n,\ \forall i\in I,n\in N \tag{31}$$

定义 9 节点-VNF 吸引度(NA,Node-VNF Attraction degree). 节点-VNF 吸引度由 t 时刻节点负载强度 $W_n(t)$和 VNF 介数权重$\mathrm{NBW}_n^i(t)$共同决定,代表 VNF 分配时趋向于负载强度相对较低且 VNF-介数权重较高的节点. 计算公式为

$$\mathrm{NA}_n^i(t)=\frac{\overline{W}_N(t)\cdot\mathrm{NBW}_n^i(t)}{W_n(t)} \tag{32}$$

当 $\Delta x_i(t+1)>0$ 时,需要增加 VNF. BCGOD 算法首先依据 t 时刻 Path($\Delta\bar{j}(t)$)及其 VNF 序列$\beta_{\Delta\bar{j}}(i)$更新节点-VNF 介数权重,结合节点负载强度得出节点-VNF 吸引度. 在资源允许的前提下优先将 VNF 分配到节点-VNF 吸引度大的节点中. 记录分配位置信息 map(i_k,n)并更新该节点资源余量和$\mathrm{NBW}_n^i(t)$,迭代此过程直到新增 VNF 分配完毕.

当 $\Delta x_i(t+1)<0$ 时,需要释放部分 VNF,为避免流量波动引起的 VNF 频繁增删,选取负载率最低的 $\Delta x_i(t+1)$个 i 型 VNF 进入 idle 状态,并将其处理的流量纳入新增 SFC 集合于下一时刻重路由. 为每个 idle 状态 VNF 设定删除缓冲时间 t_{buff},若在 t_{buff}内 idle 状态 VNF 所在底层节点上需要新增 i 型 VNF,则优先将 idle 状态 VNF 恢复为原状态,减少 VNF 部署次数;若在 t_{buff}到期时该节点上没有新增 VNF,则释放该 idle 状态 VNF. 依据 ski-rental 算法,当 VNF 进入 idle 状态时,t_{buff}由式(33)确定,由 ski-rental 算法得出 VNF 运行与部署开销之和不超过离线最优决策的 $e/(e-1)$倍[23].

$$P\{t_{\mathrm{buff}}=s\}=\left(\frac{\eta_i-1}{\eta_i}\right)^{\eta_i-s}\frac{1}{\eta_i[1-(1-1/\eta_i)^{\eta_i}]},\eta_i=\frac{\varepsilon_i}{\mu_i} \tag{33}$$

依据 map(i_k,n)得出新增的 VNF$\Delta x_i^n(t+1)$,结合 t 时刻底层网络中残留处理数据能力的 VNF 资源 $x_{ni}^{\mathrm{res}}(t)$和链路带宽,可得“虚拟资源网络”,记为 $G_{\mathrm{VRN}}(t+1)=$

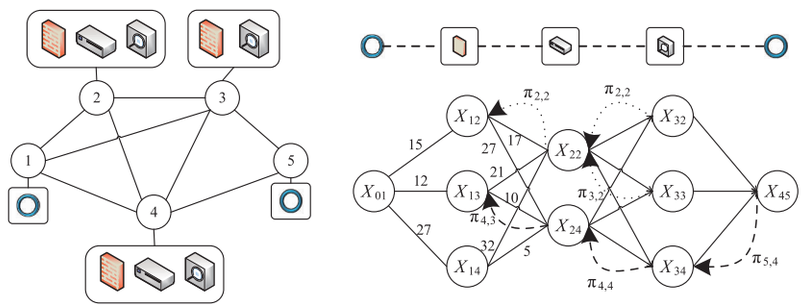
$(N_{\mathrm{VNF}}(t+1),E_{\mathrm{res}}(t+1))$. $G_{\mathrm{VRN}}(t+1)$构建伪代码如算法 1 所示,算法复杂度为 $O(\bar{J}\cdot k)=O(n^2\cdot k)$,其中 $\bar{J}$ 代表 HSFC 数目,$k=\max\{\Delta x_i(t+1)\}$.

算法 1　虚拟资源网络构建算法

输入:$G=(N,E)$,$\Delta x_i(t+1)$,$\mathrm{Path}(\Delta\bar{J}(t))$
输出:$G_{\mathrm{VRN}}(t+1)=(N_{\mathrm{VNF}}(t+1),E_{\mathrm{res}}(t+1))$
初始化 $\mathrm{map}(I,N)=\mathrm{null}$,$\mathrm{NBW}_n^i=0$
获取底层网络信息 $x_i^n(t)$,$x_{ni}^{\mathrm{res}}(t)$,$B_j^e(t)$
删除的 t_{buff}到期的 idle 状态 VNF
for　$n:N$　do
　　由 $x_i^n(t)$,C_i^{CPU} 获取 $W_n(t)$与 $\bar{W}_N(t)$
for　$i:I$　do
　　if　$\Delta x_i(t+1)<0$
　　负载最低的 $\Delta x_i(t+1)$个 VNF 进入 idle 状态,设定 t_{buff}
　　标记 idle 状态 VNF 内的 SFC 为 $t+1$ 时刻新增 SFC
　　if　$\Delta x_i(t+1)>0$
　　依据 $\mathrm{Path}(\Delta\bar{J}(t))$计算$\mathrm{NBW}_n^i$
　　更新$\mathrm{NA}_n^i(t)$,初始化 $k=0$,$K=0$
　　while $(k<\Delta x_i(t+1))$　do
　　$n=\max(\mathrm{NA}_n^i(t))$
　　if　$[C_n^{\mathrm{CPU}}-\sum_{i\in I}x_i^n(t)\cdot C_i^{\mathrm{CPU}}]>C_i^{\mathrm{CPU}}$
　　部署或恢复 idle VNFi_k 于节点 n
　　记录 $\mathrm{map}(i_k,n)$
　　更新节点 n 的资源与 $W_n(t)$
　　$K=\mathrm{Path}(\Delta\bar{j}(t))$ though n
　　for node(m) of K do
　　NBW_m^i-1
　　$k++$
　　更新 n 的$\mathrm{NA}_n^i(t)$
　　由 $\mathrm{map}(I,N)$获取 $\Delta x_i^n(t+1)$
　　$n_{\mathrm{VNF}}^i(t+1)=\Delta x_i^n(t+1)+x_{ni}^{\mathrm{res}}(t)$得出 $N_{\mathrm{VNF}}(t+1)$
　　$e_{\mathrm{res}}(t+1)=B_e-\sum_{j\in J(t)}v_j^e\alpha_j^e(t)$得出 $E_{\mathrm{res}}(t+1)$
　　$G_{\mathrm{VRN}}(t+1)=(N_{\mathrm{VNF}}(t+1),E_{\mathrm{res}}(t+1))$

3.2.2　新增 SFC 路径构建策略

新增 SFC 路径构建策略则在 $G_{\mathrm{VRN}}(t+1)$的基础上为新增 SFC 构建路径 $\mathrm{Path}(\Delta\bar{J}(t+1))$. 首先将 $\Delta\bar{J}(t+1)$中的新增 SFC 按照 $\Delta\alpha_j^*(t+1)$大小升序排列,优先部署流量较大的新增 SFC. 对于一条新增 SFC$\Delta\bar{j}$,依据$\beta_{\Delta\bar{j}}(i)$所需的 VNF 在 $G_{\mathrm{VRN}}(t+1)$中的分布将物理节点依据每层状态进行分割,构建“状态分布图”,过程如图 1 所示. “状态分布图”每层状态 X_I 中的节点 X_{IN},代表$\beta_{\Delta\bar{j}}(i)$中 VNF 在 $G_{\mathrm{VRN}}(t+1)$中的分布,如 Firewall 分布在 2,3,4 节点,记为 X_{12},X_{13},X_{14}. 源节点与目的节点分别表示为状态 X_{0s},$X_{|\beta_{\Delta\bar{j}}(i)|+1,o}$. “状态分布图”的有向边的权重代表两节点间最小传输代价,新增 SFC 基于 Viterbi 算法寻找从 X_0 到 $X_{|\beta_{\Delta\bar{j}}(i)|+1}$的最短路径. 以 SFC:{Firewall→Encryption→IDS}为例,从源节点 S 出发,当到达 Encryption 状态 X_2 时,共有 3×2 种状态转移途径,若到达 X_{24},则 3 种方式中 $\delta(X_{01},X_{13})+\delta(X_{13},X_{24})$代价最小,则建立回溯指针 $\pi_{4,3}$,代表从 X_{01} 到 X_{24} 最短路径经过 X_{13},同理对于 X_{22} 建立 $\pi_{2,2}$. 迭代计算到目的节点 X_{45},沿各节点对应的回溯指针 $\pi_{5,4}$,$\pi_{4,4}$,$\pi_{4,3}$可建立路径$\mathrm{Path}_{\Delta\bar{j}(t+1)}=\{1\to3\to4\to4\to5\}$,其中 Firewall 位于节点 3,Encryption 与 IDS 位于节点 4. 路径构建策略算法复杂度为 $\theta(N^2\cdot|\beta(i)|)$,$N$ 为 $G_{\mathrm{VRN}}(t+1)$中节点数目.



HSFC 所需VNF 在虚拟资源网络中的分布　　HSFC 对应的“状态分布图”

图1　HSFC状态分布图及其路径构建

4　仿真分析

4.1　仿真数据与环境

(1)仿真数据

实验选择马萨诸塞大学校园网流量数据集[25],使用校园网连续 55 小时发起的 YouTube 访问 request 模拟 SFC. 将每条 request 视为 SFC,为其随机生成长度为 2~4 的 VNF 序列 β_j,依据实验拓扑规模划分 22 个网段,同一网段视为相同节点,并将源、目的 IP 网段相同、β_j 相同的 SFC 合并构成 HSFC,共计 462 种. 所有 HSFC 流量构成 NFV 系统工作负载,变化趋势如图 2 所示,负载峰值约 40Gbps,PMR(Peak-to-Mean Ratio)值为 1.64,可见 0~400min,1100~1800min,2700min-3300min 时段工作负载较高且波动明显,其余时间工作负载处于上

升或下降时段,波动较小.

(2)拓扑结构

实验拓扑采用 Geant 网络. 包含 22 个节点和 36 条链路,其中 12 个节点包含 96 个 CPU,RAM 120GB,HDD 2000GB,其余节点作为转发节点,每条链路带宽 3Gbps,链路权重 ω_e 由 36 到 200 随机生成,数据中心内部传输权重由 5 到 20 随机生成.

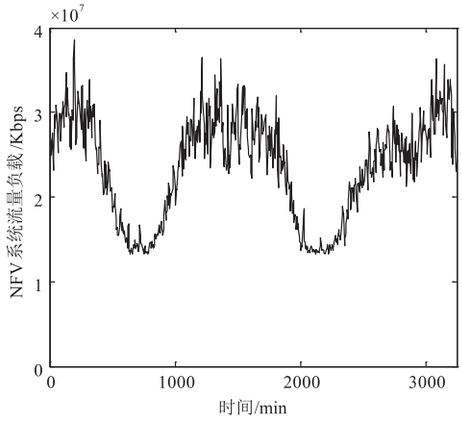


图2 NFV系统工作负载

(3)VNF 参数

实验使用 4 种 VNF 参数如表 1 所示.

表 1 VNF 参数

VNF	CPU (Core)	RAM (GB)	HDD (GB)	吞吐量 (Kbps)	增衰比 λ
Firewall	4	4	20	600000	0.9
Encryption	4	4	20	480000	1.2
IDS	8	8	40	600000	0.8
NAT	2	2	20	400000	1

(4)系统参数与实验环境

综合流量变化趋势与 VNF 部署间隔取值[11],FEPA 算法感知时间间隔取 $\Delta t = 5\text{min}$,VNF 单位时间运行开销 $\mu_i = 0.6C_i^{\text{CPU}}$ 元,VNF 部署开销取决于其运行开销[24],取 $\varepsilon_i = 6\mu_i$,私有服务器集群处理流量代价约是 VNF 处理开销的 3 倍,即 $\psi = 3\sum\mu_i/\sum B_i$,单位流量传输代价 $\delta_e = 3.6265\omega_e \times 10^{-11}$/Kbps. 仿真实验环境为 CPU:Inter Core i7-4790 3.60GHz;内存:8GB;系统:Linux 个人电脑.

4.2 实验分析

实验分为流量演化感知和在线弹性编排两个部分.

4.2.1 流量演化感知效果

图 3 对比了 FEPA 算法使用迭代更新的学习率 η_t 与使用恒定学习率 $\eta = \dfrac{\theta}{G\sqrt{T}}$ 感知流量演化效果的区别. 当学习率 η 取定值时,FEPA 算法等价于在线梯度下降法[10](OGD,Online Gradient Descent),可见模型使用恒定学习率 η 时,随时间的推移 regret 值逐渐大于更新学习率 η_t. 造成这一现象的原因是使用递减学习率 η_t 的 FEPA 算法相对于 OGD 能有效避免短期突发流量对演化感知带来的影响,拥有更稳定的在线感知性能.

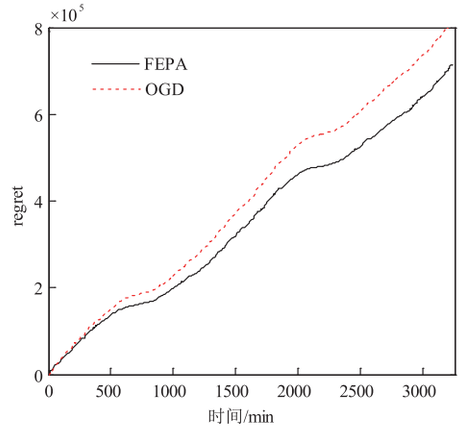


图3 学习率η_t对算法的影响

图 4 展示了不同在线学习算法的 $\lim\limits_{T\to\infty}\dfrac{\text{reg}_T}{T}$ 值与 FEPA 算法对应上界极限随时间变化的趋势,可见 FEPA 算法满足次线性增长条件,且流量演化感知效果优于其他算法. FTL 算法因缺失正则项导致 reg_T 难以收敛;OGD 算法具备收敛特性,但恒定的学习率使 OGD 的感知效果易受短期突发流量影响;VPCM 算法的感知损失函数 $f_t[\alpha_j^*(t)] = |\alpha_j^*(t) - \alpha_j(t)|$,由于 $g_{jt} = \partial f_t[\alpha_j^*(t)] = \pm 1$,造成了不论预测值与真实值的差距 $|\alpha_j^*(t) - \alpha_j(t)|$ 多大,VPCM 代理函数都只能用 ±1 进行调整,所以 $\lim\limits_{T\to\infty}\dfrac{\text{reg}_T}{T}$ 易受流量变化幅度影响.

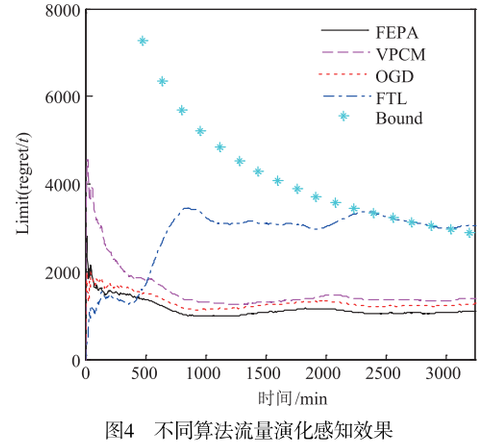


图4 不同算法流量演化感知效果

4.2.2 在线弹性编排效果

图 5 比较了 OEOP 与 VNF-OP[4],Kariz[6] 算法处理

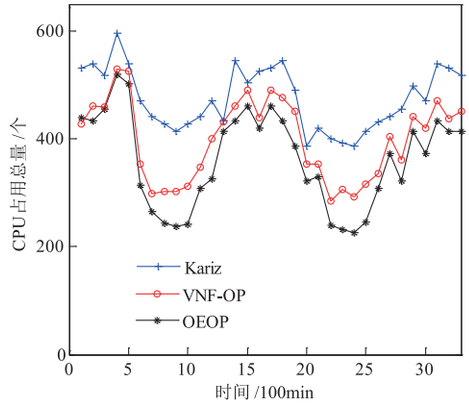


图5　不同策略占用CPU总数量对比

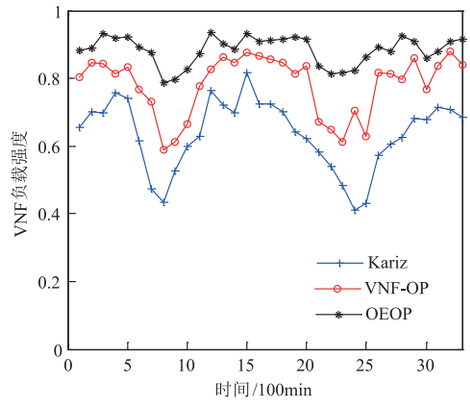


图6　不同策略VNF平均负载强度

相同工作负载时使用 CPU 总量,其中 VNF-OP,Kariz 为静态反应式策略. 可见 OEOP 使用的 CPU 总量能随工作负载变化及时调整,原因是 OEOP 基于 FEPA 算法依据工作负载预先确定了每个时隙需要增删的 VNF 数量,实时调整 VNF 供应;VNF-OP 注重 SFC 路径规划与 VNF 部署位置,没有确定最优 VNF 数量,但仍有一定程度的弹性伸缩能力;Kariz 力求构建 SFC 分布式传输路径,依据分配到节点的流量大小和 VNF 吞吐量使用装箱算法确定 VNF 数量,该算法在构建低传输代价路径上有明显优势,但因使用装箱算法求解 VNF 数量降低了 VNF 吞吐量利用率,导致该算法虚拟资源弹性伸缩效果不佳.

图 6 比较了三种算法 VNF 平均负载强度的变化. 可见低流量时段三种部署算法的 VNF 负载强度均存在一定程度的下降,但 OEOP 的 VNF 负载强度能够维持在相对稳定的水平,这是因为 NFV 系统工作负载降低时,BCGOD 子算法能及时释放了负载强度较低的 VNF,保证了 VNF 吞吐量利用率;VNF-OP 在构建路径时优先利用已部署的 VNF 资源,具备提高虚拟资源利用率的效果;Kariz 算法为保证流量传输代价优先构建分布式路径,导致其多个底层节点中的 VNF 负载较低,难以实现低负载虚拟资源的释放. 相比之下,OEOP 的 VNF 吞吐量利用率平均提升 10.2% 与 24.8%.

图 7 比较了 OEOP, GNFC[13] 算法相对 GFM 算法[14]在处理动态工作负载时的 VNF 数量减幅比. 可见 OEOP 相对 GFM 能减少 19.5% ~30.3% 的 VNF 数量,且减幅比受工作负载变化的影响较小,这是因为 FEPA 子算法能够基于工作负载变化在每个时隙确定细粒度的 VNF 弹性需求,并结合 BCGOD 算法及时释放冗余虚拟资源. GNFC 通过合并分布于不同节点的网络功能有效减小了 VNF 数量,但忽略了单个 VNF 往往存在最大处理能力约束[4,6],如表 2 所示的最大吞吐量. 若合并于同一节点的网络功能所需吞吐量超出已有 VNF 的最大吞吐量约束,则需要继续增加 VNF 以容纳超出部分. 此外,在 NFV 系统工作负载较低的时段,GNFC 的减幅比下降较为明显,该现象与文献[13]的结论是一致的. 综合上述两种因素的影响,GNFC 算法只能减少 12.2% ~22.8% 的 VNF 数量.

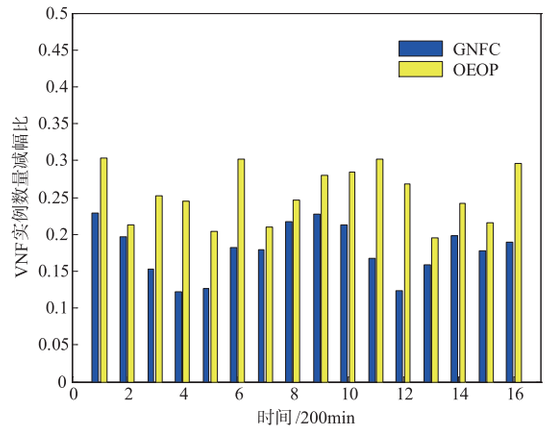


图7　VNF数量相对GFM算法减幅比

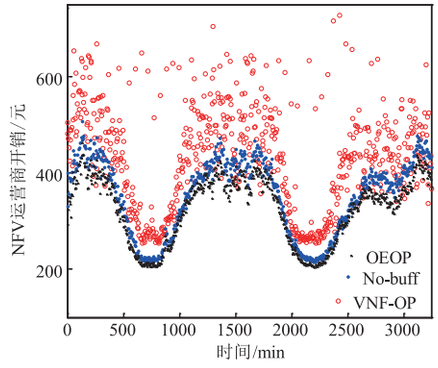


图8　NFV运营商每时刻对应的开销

OEOP 与 VNF-OP、No-buff 三种算法得出的 NFV 运营商开销对比效果分别如图 8 所示. 其中 No buff 在 OEOP 算法的基础上,不对 VNF 设置删除缓存 t_{buff},与

OEOP 相比,在 NFV 系统工作负载波动较大的时段 NFV 运营商开销略高,而在工作负载持续增长或降低的时段,开销差别不明显,说明设置 t_{buff} 能有效减缓流量波动引起的 VNF 频繁增删以降低 $D_{\text{deploy}}(t)$. VNF-OP 算法作为静态反应式策略在每个时隙重复运行,得出互不相关的部署策略,导致 VNF 部署位置相对前一时刻可能发生明显变动,增加了 VNF 迁移引起的部署代价 $D_{\text{migrate}}(t)$. 采用 OEOP 策略相对 VNF-OP 算法能帮助 NFV 运营商平均降低 26.7% 的开销.

5 结论

本文提出一种基于流量演化感知的 SFC 在线弹性编排策略. 该策略借助在线学习算法进行流量演化感知,预先确定了细粒度的 VNF 增量,并以 SFC 路径与底层节点负载两因子为导向在线决策 VNF 部署. 仿真表明该策略明显增强了虚拟资源供求匹配特性,提高了 VNF 利用率,避免了 VNF 迁移造成的部署代价,有效降低了 NFV 运营商开销. 后续将结合弹性编排与强化学习进行研究,提高网络资源自适应性.

参考文献

[1] GEMBER-J A, VISWANATHAN R, PRAKASH C, et al. OpenNF: Enabling innovation in network function control [A]. ACM Conference on Sigcomm [C]. Chicago, Illinois: ACM, 2014. 163 – 174.

[2] QUINN. E. P, NADEAU. E. T, Problem Statement for Service Function Chaining [R]. New York: IETF, RFC, 2015. 8 – 10.

[3] MOENS H, TURCK F D. VNF-P: A model for efficient placement of virtualized network functions [A]. International Conference on Network & Service Management (CNSM) [C]. Rio de Janeiro, Brazil: IEEE, 2014. 418 – 423.

[4] BARI F, CHOWDHURY S R, AHMED R, et al. Orchestrating virtualized network functions [J]. IEEE Transactions on Network & Service Management, 2016, 13 (4): 725 – 739.

[5] JANG I, SUH D, PACK S, et al. Joint optimization of service function placement and flow distribution for service function chaining [J]. IEEE Journal on Selected Areas in Communications, 2017, 35(11): 2532 – 2541.

[6] GHAZNAVI M, SHAHRIAR N, KAMALI S, et al. Distributed service function chaining [J]. IEEE Journal on Selected Areas in Communications, 2017, 35(11): 2479 – 2489.

[7] COHEN R, LEWIN E. L, NAOR J S, et al. Near optimal placement of virtual network functions [A]. IEEE Conference on Computer Communications (INFOCOM) [C]. Hong Kong, China: IEEE, 2015. 1346 – 1354.

[8] WANG X, WU C, LE F, et al. Online VNF Scaling in Datacenters [A]. IEEE International Conference on Cloud Computing [C]. San Francisco, USA: IEEE, 2016. 140 – 147.

[9] ARTEAGA C H T, RISSOI F, RENDON OMC. An adaptive scaling mechanism for managing performance variations in network functions virtualization: Acase study in an NFV-based EPC [A]. International Conference on Network and Service Management (CNSM) [C]. Tokyo, Japan: IEEE, 2017. 1 – 7.

[10] ZHANG X, WU C, Li Z, et al. Proactive VNF provisioning with multi-timescale cloud resources: Fusing online learning and online optimization [A]. IEEE Conference on Computer Communications (INFOCOM) [C]. Atlanta, USA: IEEE, 2017. 1 – 9.

[11] FEI X, LIU F, XU H, et al. Adaptive VNF scaling and flow routing with proactive demand prediction [A] IEEE Conference on Computer Communications (INFOCOM) [C]. Honolulu, USA: IEEE, 2018. 486 – 494.

[12] WANG M, MENG X, ZHANG L. Consolidating virtual machines with dynamic bandwidth demand in data centers [A]. IEEE Conference on Computer Communications (INFOCOM) [C]. Shanghai, China: IEEE, 2011. 71 – 75.

[13] Wen T, Yu H, Sun G, et al. Network function consolidation in service function chaining orchestration [A]. IEEE International Conference on Communications (ICC) [C]. Kuala Lumpur, Malaysia: IEEE, 2016. 1 – 6.

[14] MIJUMBI R, SERRAT J, GORRICHO J L, et al. Design and evaluation of algorithms for mapping and scheduling of virtual network functions [A]. Proceedings of the First IEEE Conference on Network Softwarization [C]. London, UK IEEE, 2015.

[15] 陈卓,冯钢,刘蓓,等. 运营商网络中面向时延优化的服务功能链迁移重配置策略[J]. 电子学报,2018,46(9): 2229 – 2237.
CHEN Z, FENG G, LIU B, et al. Delay optimization oriented service function chain migration and re-deployment in operator network [J]. Acta Electronica Sinica, 2018, 46 (9): 2229 – 2237. (in Chinese)

[16] JIA Y, WU C, LI Z, et al. Online scaling of nfv service chains across geo-distributed datacenters [J]. IEEE/ACM Transactions on Networking, 2016, 26(2): 699 – 710.

[17] CHAPARADZA R, CIAVAGLIA L, WŬDCZAK M, et al. ETSI Industry Specification Group on Autonomic Network Engineering for the Self-managing Future Internet (ETSI ISG AFI) [M]. Poznan Poland: Web Information Systems Engineering (WISE) 2009. 622 – 638.

[18] ETSI GS, Network Functions Virtualisation (NFV), Pre-

deployment Testing[R]. London:ETSI,2012. 15 – 22.
[19] MCMAHAN H B, Holt G, SCULLEY D, et al. Ad click prediction: a view from the trenches[A]. International Conference on Knowledge Discovery and Data mining[C]. Chicago, USA: ACM, 2013. 1222 – 1230.
[20] CHEN N, AGARWAL A, WIERMAN A, et al. Online convex optimization using predictions[J]. ACM Sigmetrics Performance Evaluation Review, 2015, 43(1): 191 – 204.
[21] SHALEV-SHWARTZ S. Online learning and online convex optimization[J]. Foundations & Trends in Machine Learning, 2012, 4(2): 107 – 194.
[22] MCMAHAN H B. A survey of algorithms and analysis for adaptive online learning[J]. The Journal of Machine Learning Research, 2017, 18(1): 3117 – 3166.
[23] KARLIN A R, MANASSE M S, MCGEOCH L A, et al. Competitive randomized algorithms for nonuniform problems[J]. Algorithmica, 1994, 11(6): 542 – 571.
[24] LIN M, WIERMAN A, ANDREW L L H, et al. Dynamic right-sizing for power-proportional data centers[J]. IEEE/ACM Transactions on Networking (TON), 2013, 21(5): 1378 – 1391.
[25] ZINK M, SUH K, GU Y, et al. Characteristics of YouTube network traffic at a campus network-measurements, models, and implications[J]. Computer Networks, 2009, 53(4): 501 – 514.

作者简介

谷允捷(通讯作者)　男,1994 年生,山东济宁人. 国家数字交换系统工程技术研究中心硕士研究生,研究方向为网络功能虚拟化.
E-mail:lizardwhite@163.com

胡宇翔　男,1982 生,河南周口人. 国家数字交换系统工程技术研究中心副研究员、博士生导师,研究方向为新型网络体系结构与核心技术.

丁悦航　女,1995 年生,山东济南人. 国家数字交换系统工程技术研究中心硕士研究生,研究方向为知识图谱.

谢记超　男,1993 年生,河北石家庄人. 国家数字交换系统工程技术研究中心硕士研究生,研究方向为网络安全与虚拟化技术.