(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 115913952 B
(45)授权公告日 2023.08.01

(21)申请号 202211352904.1
(22)申请日 2022.11.01
(65)同一申请的已公布的文献号
申请公布号 CN 115913952 A
(43)申请公布日 2023.04.04
(73)专利权人 南京航空航天大学
地址 210016 江苏省南京市秦淮区御道街29号
(72)发明人 王然 余雪
(74)专利代理机构 南京合砺专利商标代理事务所(普通合伙) 32518
专利代理师 鲍小龙
(51)Int.Cl.
H04L 41/0823(2022.01)
H04L 41/142(2022.01)
H04L 41/0895(2022.01)
审查员 王英

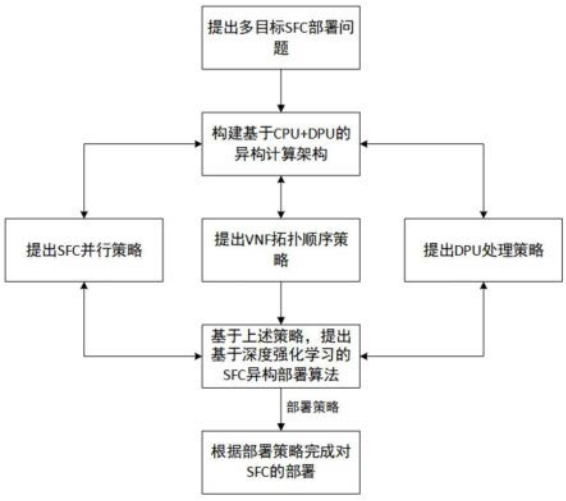
权利要求书4页 说明书12页 附图7页

(54)发明名称

基于CPU+DPU平台的多目标服务功能链的高效并行化和部署方法

(57)摘要

本发明公开了一种基于CPU+DPU平台的多目标服务功能链的高效并行化和部署方法，主要用于解决当前传统计算架构和传统SFC部署系统面临的转发性能下降、场景和需求多样化等问题。所述方法主要通过构建一个异构计算架构来解决多目标部署问题，该架构由编排器和基于CPU+DPU结构的服务器组成，编排器负责接收来自网络运营商的SFC请求，运行基于深度强化学习的SFC部署算法包括并行策略、VNF拓扑顺序策略和DPU处理策略以获得每个请求的最佳部署方案，然后调用资源管理模块来管理资源，最后调用驱动模块将部署方案传递给放置的服务器，由服务器根据部署方案分别使用CPU或DPU完成对SFC的部署。



CN 115913952 B

1.一种基于CPU+DPU平台的多目标服务功能链的高效并行化和部署方法,其特征在于:包括如下步骤:

(1)构建一个异构计算架构用于解决多目标部署问题,所述的异构计算架包括一个负责管理和协调的编排器和基于CPU+DPU结构的服务器;

(2)依据VNF间的独立性,通过SFC并行策略将串行SFC转化为并行SFC,据此解决SFC并行问题;

所述的SFC并行问题的目标函数是最小化由数据包复制和合并引起的额外的资源消耗和时延:

min(αC+βΔd)

其中α和β分别表示额外的资源消耗和时延的权重系数,$C=\sum_{\mu=1}^{|R|}\sum_{\forall B\in r_\mu}64\times\frac{\Phi_B-1}{U}$表示数据包复制和合并引起的额外资源消耗,$\Delta d=\sum_{\mu=1}^{|R|}30\times\frac{V_\mu}{10^9}$表示数据包复制和合并引起的额外时延;R表示网络中的一组SFC请求,r_μ表示第μ个SFC请求,B是一组并行分支,Φ_B表示B的并行度,U表示数据包的大小,V_μ是请求R中第μ个SFC的数据量;

所述的SFC并行策略目的是根据VNF之间的依赖关系识别链中的VNF以找到所有可以并行执行的链;

其中SFC并行策略包括将VNF分成monitors和shapers两种类型,其中monitors负责监测流量而不作任何修改,而shapers用来处理和修改流量,具体算法流程如下:

21)初始化分支链集合B、主链S,monitor集合M;

22)遍历r_μ,如果$f_i^\mu\in r_\mu$为monitor,首先初始化一个分支链b∈B,然后将f_i^μ加入b以及M中;如果$f_i^\mu\in r_\mu$为shaper,则将其加入主链S中,此时在M中搜索f_i^μ所依赖的monitor,对于每一个这样的monitor,比如k∈M,有一个目前以k为终点的分支链,然后将k指向f_i^μ,从而将分支链扩展到以f_i^μ为终点,并从M中移除k;

23)调用路径搜索算法找到所有可以并行执行的路径集合PATH;

24)返回分支链集合B、主链S以及路径集合PATH;

(3)针对某一时刻多条SFC同时到达的情况,采用VNF拓扑顺序算法,按照该算法得到的拓扑顺序部署VNF,并结合VNF的共享和缩放特性降低时延;

所述的VNF拓扑顺序算法用于构建一个关于若干条SFC的VNF拓扑顺序,按照该顺序部署VNF实现降低时延;

所述VNF拓扑顺序算法的具体流程如下:

31)初始化f为源节点;

32)遍历同一时刻到达的请求集合R,调用SFC并行策略得到分支链集合B和主链S以及路径集合PATH,并根据路径集合PATH求得所有路径中时延最大的路径,将其加入到集合C中;

33)遍历C,创建有向加权图graph=(F,ω),F为VNFs的集合,ω为边与边之间的权重;

34)调用最小反馈弧集算法求graph的最小反馈弧集,并验证所求拓扑顺序是否满足不同链间VNF的依赖关系,如果满足则返回拓扑顺序,否则返回False;

(4)针对实时性要求高的服务请求采用DPU处理策略,所述DPU处理策略用于处理包括网络协议处理、数据加解密、数据压缩在内的计算任务,以节省CPU计算资源并减少时延;

如果某一时刻到达的请求集中有高优先级请求,判断该请求中是否有负责该数据处理任务的VNF,若有则使用DPU进行快速处理,若没有,则使用CPU进行处理;

(5)提出基于深度强化学习的SFC异构部署算法,该异构部署算法能够针对不同时刻到达请求的数量和情况分别完成对SFC的部署;

所述的基于深度强化学习的SFC异构部署算法针对不同时刻到达请求的数量和情况分别采用不同的策略进行处理,包括执行并行策略、VNF拓扑顺序策略和DPU处理策略,以此实现对SFC的部署;

步骤(5)中SFC的部署的具体操作如下:

51)系统先删除超时请求,经过优先级判断器将到达的请求R按照实时性进行划分,实时性高的划分为高优先级R_high,实时性低的划分为低优先级R_low;

52)初始化时隙τ;

53)根据R_high和R_low的数量决定采用何种策略对SFC进行处理;

54)构建神经网络模型并训练,将当前物理网络的状态和正在处理的请求的特征作为输入,通过神经网络的计算,输出每个VNF的部署策略;

55)更新网络状态。

2.根据权利要求1所述的基于CPU+DPU平台的多目标服务功能链的高效并行化和部署方法,其特征在于:步骤(1)所述的异构计算架构由编排器和服务器组成,其中编排器包括SFC部署算法模块,资源管理模块以及驱动模块;

所述服务器包括CPU、DPU以及连接CPU和DPU的总线;

所述编排器负责管理和部署到达的SFC,由异构计算结构组成的服务器则负责根据编排器传达的部署策略依次处理不同SFC中的VNF;其中编排器的具体任务包括接收来自网络运营商的SFC请求,运行SFC部署算法以确定哪些SFC被接受以及这些被接受的SFC如何被放置;

针对不同请求的不同情况,所述方法将分别调用并行策略、VNF拓扑顺序策略和DPU处理策略,以获得每个请求的最佳部署方案,然后调用资源管理模块来管理资源,最后调用驱动模块将部署方案传递给放置的服务器,由服务器根据部署方案分别使用CPU或DPU完成对SFC的部署。

3.根据权利要求1或2所述的基于CPU+DPU平台的多目标服务功能链的高效并行化和部署方法,其特征在于:步骤(1)所述的多目标部署问题具体描述如下:

SFC部署问题的目标函数如下:

$\min(f_1+f_2+f_3)$

$s.t.C_1,C_2,C_3$

$$\begin{cases} f_1 = \sum_{r_\mu \in R} D_\mu \\ f_2 = \sum_{r_\mu \in R} y_{r_\mu} B_\mu \tau_r \\ f_3 = C(\tau) \end{cases}$$

对于f_1,D_μ为总体响应时延,即存在:

$$D_\mu = L_\mu + P_\mu + T_\mu + \overline{W_q}$$

其中,L_μ为通信时延,P_μ为处理时延,T_μ为传输时延,$\overline{W_q}$为平均排队时延;

对于f_2,$\sum_{r_\mu \in R} y_{r_\mu} B_\mu \tau_r$为总吞吐量,其中二进制变量$y_{r_\mu}$表示$r_\mu$是否被接受,$B_\mu$为SFC的最小带宽,$\tau_r = 1 * \Delta$表示SFC的生存时间;

对于f_3,$C\tau)$表示总的部署成本,存在如下表达式:

$C\tau) = SC(\tau) + C_{scale}(\tau) + C_{DPU}(\tau)$

$SC(\tau)$表示总运营成本,即为服务器打开成本和成功放置VNF的成本之和:

$$SC(\tau) = \sum_{n_i \in N} \sum_{f_v^\mu \in F_\mu} x_{f_v^\mu}^{n_i} \zeta_c C_{f_v^\mu} + \sum_{e_j \in E} \sum_{e_h^\mu \in E_\mu} x_{e_h^\mu}^{e_j} \zeta_B B_\mu + \sum_{n_i \in N} \zeta_O$$

$x_{f_v^\mu}^{n_i}$表示请求$r_\mu \in R$中,VNF $f_v^\mu \in F_\mu$是否部署在服务器节点$n_i \in N$上,$x_{e_h^\mu}^{e_j}$表示请求$r_\mu \in R$中,虚拟链路$e_h^\mu \in E_\mu$是否映射到物理链路$e_j \in E$上,ζ_c和ζ_B分别表示资源和带宽的单位成本,$C_{f_v^\mu}$表示VNF $f_v^\mu \in F_\mu$的资源需求,ζ_0表示服务器打开成本;

$C_{scale}(\tau) = \sum_{n_i \in N} \sum_{f_v^\mu \in F_\mu} C^h \cdot x_{n_i,h}^{f_v^\mu}$表示总缩放成本,其中$C^h$表示水平缩放的成本,$x_{n_i,h}^{f_v^\mu}$表示VNF $f_v^\mu \in F_\mu$是否进行水平缩放;

$C_{DPU}(\tau)$表示DPU的总使用成本,定义如下:

$$C_{DPU}(\tau) = \sum_{n_i \in N} \sum_{f_v^\mu \in F_\mu} \zeta_{c_D} x_{n_i,D}^{f^\mu} C_{f_v^\mu} + \zeta_{B_D} x_{n_i,D}^{f_v^\mu} B_\mu$$

其中,ζ_{c_D}和ζ_{B_D}使用DPU时资源和带宽的单位成本,$x_{n_i,D}^{f^\mu}$表示VNF $f_v^\mu \in F_\mu$是否使用DPU进行处理;

C_1表示资源约束,C_2表示带宽约束,C_3表示延时约束。

4.根据权利要求1所述的基于CPU+DPU平台的多目标服务功能链的高效并行化和部署方法,其特征在于:步骤(2)所述的SFC并行问题的具体描述如下:

针对部分独立工作而不影响其他VNF的串行SFC转换为并行SFC;由于SFC中的VNF必须按照特定的顺序应用于每个数据包流,因此它们形成了一种依赖关系;所述方法中规定,如果一个VNF f_v^μ在另一个VNF f_{v+1}^μ之前,称f_{v+1}^μ依赖于f_v^μ,记为$f_v^\mu \prec f_{v+1}^\mu$;

为了并行地处理数据包,需要复制和合并两个功能,当一个数据包进来时,复制功能会复制该数据包并将其发送给可以并行处理的VNF,在数据包被处理后,复制的数据包被合并功能合并,数据包复制和合并会引起额外的资源消耗和时延。

5.根据权利要求1所述的基于CPU+DPU平台的多目标服务功能链的高效并行化和部署方法，其特征在于：对于SFC并行问题，引入流量约束，用0_c表示r_μ中复制节点的集合，0_m表示r_μ中合并节点的集合，$0_c n_i$和$0_m n_i$分别表示r_μ中复制节点和合并节点的数量；对于SFCr_μ，除了复制节点、合并节点、源节点和目的节点不满足流量守恒外，所有中间节点都满足流量守恒。

基于CPU+DPU平台的多目标服务功能链的高效并行化和部署方法

技术领域

[0001] 本发明属于服务功能链编排技术,具体涉及一种基于CPU+DPU平台的多目标服务功能链的高效并行化和部署方法。

背景技术

[0002] 传统基于中央处理单元(CPU)完成数据转发的模式出现瓶颈,仅由CPU组成的计算架构已无法满足多样化的场景和业务需求。一方面,由于摩尔定律放缓,网络带宽和连接数的剧增比数据的通路更宽、更密,直接将处于端、边、云各处的计算节点暴露在了剧增的数据量下,CPU的算力增长速度与网络带宽的增长速度呈现“剪刀差”;另一方面,在高并发转发工作中,CPU的串行计算模式难以发挥最大计算能力。

[0003] 网络功能虚拟化(NFV)的出现,为设计、协调、部署和规范各种移动服务以支持日益复杂和多变的服务请求提供了一种新颖的方式,使SFC的部署更加灵活和敏捷。目前的服务功能链(SFC)部署系统专注于优化网络资源利用率,没有考虑到服务需求的多样性,以及服务转发性能的下降。

[0004] 基于数据处理单元(DPU)的云架构按需卸载和加速网络和存储系统,与图形处理单元(GPU)、现场可编程门阵列(FPGA)和特定应用集成电路(ASIC)相比,解放了更高成本的主机CPU算力,可以更经济有效地获得极致性能。DPU引发的架构变革一定程度上提升了整个数据中心资源池的能效成本比和公有云厂商的收益成本比,未来DPU可以覆盖越来越多的需求和场景,对此,基于DPU解决当前传统计算架构和传统SFC部署系统面临的一些问题是本发明研究内容。

发明内容

[0005] 发明目的:为了解决当前传统计算架构和传统SFC部署系统面临的转发性能下降、场景和需求多样化等问题,本发明提供一种基于CPU+DPU平台的多目标服务功能链的高效并行化和部署方法。

[0006] 技术方案:一种基于CPU+DPU平台的多目标服务功能链的高效并行化和部署方法,包括如下步骤:

[0007] (1)构建一个异构计算架构用于解决多目标部署问题,所述的异构计算架包括一个负责管理和协调的编排器和基于CPU+DPU结构的服务器;

[0008] (2)依据虚拟网络功能(VNF)间的独立性,通过SFC并行策略将串行SFC转化为并行SFC,据此解决SFC并行问题;

[0009] 所述的SFC并行问题的目标函数是最小化由数据包复制和合并引起的额外的资源消耗和时延:

[0010] $\min(\alpha C+\beta \Delta d)$

[0011] 其中α和β分别表示额外的资源消耗和时延的权重系数，$C=\sum_{\mu=1}^{|R|}\sum_{\forall B\in r_\mu}64\times\frac{\Phi_B-1}{U}$表示数据包复制和合并引起的额外资源消耗，$\Delta d=\sum_{\mu=1}^{|R|}30\times\frac{V_\mu}{10^9}$表示数据包复制和合并引起的额外时延；

[0012] 所述的SFC并行策略目的是根据VNF之间的依赖关系识别链中的VNF以找到所有可以并行执行的链；

[0013] (3)针对某一时刻多条SFC同时到达的情况，采用VNF拓扑顺序算法，按照该算法得到的拓扑顺序部署VNF，并结合VNF的共享和缩放特性降低时延；

[0014] 所述的VNF拓扑顺序算法用于构建一个关于若干条SFC的VNF拓扑顺序，按照该顺序部署VNF实现降低时延；

[0015] (4)针对实时性要求高的服务请求采用DPU处理策略，所述DPU处理策略用于处理包括网络协议处理、数据加解密、数据压缩在内的计算任务，以节省CPU计算资源并减少时延；

[0016] 如果某一时刻到达的请求集中有高优先级请求，判断该请求中是否有负责该数据处理任务的VNF，若有则使用DPU进行快速处理，若没有，则使用CPU进行处理；

[0017] (5)提出基于深度强化学习的SFC异构部署算法，该异构算法能够针对不同时刻到达请求的数量和情况分别完成对SFC的部署；

[0018] 所述的基于深度强化学习的SFC异构部署算法针对不同时刻到达请求的数量和情况分别采用不同的策略进行处理，即并行策略、VNF拓扑顺序策略和DPU处理策略，以此实现对SFC的部署。

[0019] 进一步的，步骤(1)所述的异构计算架构具体描述如下：

[0020] 该架构由编排器和服务器组成，其中编排器包括SFC部署算法模块，资源管理模块以及驱动模块；服务器包括CPU、DPU以及连接CPU和DPU的总线。编排器负责管理和部署到达的SFC，而由异构计算结构组成的服务器则负责根据编排器传达的部署策略依次处理不同SFC中的VNF。其中编排器的具体任务包括接收来自网络运营商的SFC请求，运行SFC部署算法以确定哪些SFC被接受以及这些被接受的SFC如何被放置。为了考虑不同请求的不同情况，该算法将分别调用并行策略、VNF拓扑顺序策略和DPU处理策略，以获得每个请求的最佳部署方案，然后调用资源管理模块来管理资源，最后调用驱动模块将部署方案传递给放置的服务器，由服务器根据部署方案分别使用CPU或DPU完成对SFC的部署。

[0021] 进一步的，步骤(1)所述的多目标SFC部署问题具体描述如下：

[0022] SFC部署问题的目标函数如下：

[0023] $\min(f_1+f_2+f_3)$

[0024] $s.t.C_1,C_2,C_3$

[0025] $$\begin{cases} f_1=\sum_{r_\mu\in R}D_\mu \\ f_2=\sum_{r_\mu\in R}y_{r_\mu}B_\mu\tau_r \\ f_3=C(\tau) \end{cases}$$

[0026] 对于f_1,D_μ为总体响应时延,即

[0027] $$D_\mu = L_\mu + P_\mu + T_\mu + \overline{W_q}$$

[0028] 其中$L_\mu = \sum_{e_h^\mu \in E_\mu} \sum_{e_j \in E} x_{e_h^\mu}^{e_j} D_{e_j}$为通信时延,$P_\mu = \sum_{f_v^\mu \in F_\mu} \sum_{n_i \in N} x_{f_v^\mu}^{n_i} \cdot \frac{1}{\frac{\eta_{m_i}^\mu c_{m_i}}{w_{m_i}^\mu} - \lambda_\mu + \varepsilon}$为处理时延,$T_\mu = \sum_{f_v^\mu \in F_\mu} \frac{U}{v_\mu}$为传输时延,$\overline{W_q}$为平均排队时延。

[0029] 对于f_2,$\sum_{r_\mu \in R} y_{r_\mu} B_\mu \tau_r$为总吞吐量,其中二进制变量$y_{r_\mu}$表示$r_\mu$是否被接受,$B_\mu$为SFC的最小带宽,$\tau_r = 1 * \Delta$表示SFC的生存时间。

[0030] 对于f_3,$C(\tau)$表示总的部署成本,即

[0031] $$C(\tau) = SC(\tau) + C_{scale}(\tau) + C_{DPU}(\tau)$$

[0032] $SC(\tau)$表示总运营成本,即为服务器打开成本和成功放置VNF的成本之和:

[0033] $$SC(\tau) = \sum_{n_i \in N} \sum_{f_v^\mu \in F_\mu} x_{f_v^\mu}^{n_i} \zeta_c C_{f_v^\mu} + \sum_{e_j \in E} \sum_{e_h^\mu \in E_\mu} x_{e_h^\mu}^{e_j} \zeta_B B_\mu + \sum_{n_i \in N} \zeta_0$$

[0034] $x_{f_v^\mu}^{n_i}$表示请求$r_\mu \in R$中,VNF $f_v^\mu \in F_\mu$是否部署在服务器节点$n_i \in N$上,$x_{e_h^\mu}^{e_j}$表示请求$r_\mu \in R$中,虚拟链路$e_h^\mu \in E_\mu$是否映射到物理链路$e_j \in E$上,ζ_c和ζ_B分别表示资源和带宽的单位成本,$C_{f_v^\mu}$表示VNF $f_v^\mu \in F_\mu$的资源需求,ζ_0表示服务器打开成本。

[0035] $C_{scale}(\tau) = \sum_{n_i \in N} \sum_{f_v^\mu \in F_\mu} C^h \cdot x_{n_i,h}^{f_v^\mu}$表示总缩放成本,其中$C^h$表示水平缩放的成本,$x_{n_i,h}^{f_v^\mu}$表示VNF $f_v^\mu \in F_\mu$是否进行水平缩放。

[0036] $C_{DPU}(\tau)$表示DPU的总使用成本,定义如下:

[0037] $$C_{DPU}(\tau) = \sum_{n_i \in N} \sum_{f_v^\mu \in F_\mu} \zeta_{c_D} x_{n_i,D}^{f_v^\mu} C_{f_v^\mu} + \zeta_{B_D} x_{n_i,D}^{f_v^\mu} B_\mu$$

[0038] 其中,ζ_{c_D}和ζ_{B_D}使用DPU时资源和带宽的单位成本,$x_{n_i,D}^{f_v^\mu}$表示VNF $f_v^\mu \in F_\mu$是否使用DPU进行处理。

[0039] 资源约束如下:

[0040] $$C_1: \ \forall n_i \in N, \sum_{f_v^\mu \in F_\mu} \sum_{l \in N^{f_v^\mu}} s_{n_i,\tau}^{f_v^\mu} \cdot C_{f_v^\mu} \le C_{n_i}$$

[0041] 其中,$s_{n_i,\tau}^{f_v^\mu}$表示部署在节点$n_i \in N$上的VNF $f_v^\mu \in F_\mu$的服务实例的数量,C_{n_i}表示节点$n_i \in N$的资源(CPU和memory)大小。

[0042] 带宽约束如下:

[0043] $$C_2: \ \forall e_j \in E, \sum_{r_\mu \in R} \sum_{e_h^\mu \in E_\mu} x_{e_h^\mu}^{e_j} \cdot a_{r,\tau} \cdot B_\mu \le B_{e_j}$$

[0044] 其中,$a_{r,\tau}$表示请求$r_\mu \in R$是否仍在服务当中,B_{e_j}表示节点$n_i \in N$的带宽大小。

[0045] 延迟约束如下:

[0046] C_3: $\forall r_\mu \in R, D_\mu \leq D_\mu^{max}$

[0047] 其中，D_μ^{max}表示最大最大端到端延迟大小。

[0048] 进一步的，步骤(2)所述的SFC并行问题的具体描述如下：

[0049] 在SFC中，某些VNF可以独立工作而不影响其他VNF，所以可以将串行SFC转换为并行SFC。然而，并不是SFC中的所有VNF都可以并行工作，如果当两个VNF都修改流的内容或违反依赖性约束时，两个VNF的操作是冲突的。只有当VNF在SFC中相互独立时，才能在VNF之间并行处理；否则，网络和服务策略的正确性可能会被破坏。

[0050] 本发明所述方法将VNF分成两种类型：monitors和shapers，其中monitors负责监测流量而不作任何修改，而shapers用来处理和修改流量。由于SFC中的VNF必须按照特定的顺序应用于每个数据包流，因此它们形成了一种依赖关系。本发明中规定，如果一个VNF f_v^μ在另一个VNF f_{v+1}^μ之前，称f_{v+1}^μ依赖于f_v^μ，记为 $f_v^\mu \prec f_{v+1}^\mu$。

[0051] 为了并行地处理数据包，需要两个功能，即复制功能和合并功能。

[0052] 当一个数据包进来时，复制功能会复制该数据包并将其发送给可以并行处理的VNF。在数据包被处理后，复制的数据包被合并功能合并。数据包复制和合并会引起额外的资源消耗和时延，因此在SFC并行问题中，目标函数是最小化由数据包复制和合并引起的额外的资源消耗和时延：

[0053] min(αC+βΔd)

[0054] 其中α和β分别表示额外的资源消耗和时延的权重系数。

[0055] C表示数据包复制和合并引起的额外资源消耗，公式如下：

[0056] $$C = \sum_{\mu=1}^{|R|} \sum_{\forall B \in r_\mu} 64 \times \frac{\Phi_B - 1}{U}$$

[0057] 其中B是一组并行分支，Φ_B表示B的并行度，U表示数据包的大小。

[0058] 数据包复制和合并引起的额外延迟Δd可表示为：

[0059] $$\forall B \in r_\mu \in R, \Delta d = \sum_{\mu=1}^{|R|} 30 \times \frac{V_\mu}{10^9}$$

[0060] 其中V_μ是请求R中第μ个SFC的数据量。

[0061] 对于SFC并行问题，引入流量约束。用O_c表示r_μ中复制节点的集合，O_m表示r_μ中合并节点的集合。$O_c(n_i)$和$O_m(n_i)$分别表示r_μ中复制节点和合并节点的数量。对于SFC r_μ，除了复制节点、合并节点、源节点和目的节点不满足流量守恒外，所有中间节点都满足流量守恒，即：

[0062] $$\forall r_\mu \in R, \forall n_h \in N, n_h \notin \{n_{src}, n_{dst}, O_c, O_m\}:$$

[0063] $$\sum_{e_h^\mu \in E_\mu} \sum_{n_g \in N} x_{e_h^\mu}^{n_g, n_h} - \sum_{e_h^\mu \in E_\mu} \sum_{n_k \in N} x_{e_h^\mu}^{n_h, n_k} = 0$$

[0064] 如果源节点是一个复制节点，则需要满足该约束条件：

[0065] $$\forall r_\mu \in R, \forall n_{src} \in Oc:$$

[0066] $$\sum_{e_h^\mu \in E_\mu} \sum_{n_u \in N} x_{e_h^\mu}^{n_{src},n_u} - \sum_{e_h^\mu \in E_\mu} \sum_{n_v \in N} x_{e_h^\mu}^{n_v,n_{src}} = O_c(n_{src})$$

[0067] 如果目的节点是一个合并节点,则需要满足如下约束条件:

[0068] $\forall r_\mu \in R, \forall n_{dst} \in Om:$

[0069] $$\sum_{e_h^\mu \in E_\mu} \sum_{n_v \in N} x_{e_h^\mu}^{n_v,n_{dst}} - \sum_{e_h^\mu \in E_\mu} \sum_{n_u \in N} x_{e_h^\mu}^{n_{dst},n_u} = O_m(n_{dst})$$

[0070] 对于其他节点是复制节点的情况,应该满足以下公式:

[0071] $\forall r_\mu \in R, \forall n_h \in Oc, \sum_{f_v^\mu \in F_\mu} x_{f_v^\mu}^{n_h} = 1:$

[0072] $\sum_{e_h^\mu \in E_\mu} \sum_{n_u \in N} x_{e_h^\mu}^{n_h,n_u} - \sum_{e_h^\mu \in E_\mu} \sum_{n_v \in N} x_{e_h^\mu}^{n_v,n_h} = O_c(n_h) - O_m(n_h)$

[0073] 对于其他节点是合并节点的情况,需要满足以下公式:

[0074] $\forall r_\mu \in R, \forall n_h \in Om, \sum_{f_v^\mu \in F_\mu} x_{f_v^\mu}^{n_h} = 1:$

[0075] $\sum_{e_h^\mu \in E_\mu} \sum_{n_v \in N} x_{e_h^\mu}^{n_v,n_h} - \sum_{e_h^\mu \in E_\mu} \sum_{n_u \in N} x_{e_h^\mu}^{n_h,n_u} = O_m(n_h) - O_c(n_h)$

[0076] 进一步的,步骤(2)所述的SFC并行策略目的是根据VNF之间的依赖关系识别链中的VNF以找到所有可以并行执行的链。所述SFC并行策略具体算法流程如下:

[0077] 21)初始化分支链集合B、主链S,monitor集合M;

[0078] 22)遍历r_μ,如果$f_i^\mu \in r_\mu$为monitor,首先初始化一个分支链b∈B,然后将f_i^μ加入b以及M中;如果$f_i^\mu \in r_\mu$为shaper,则将其加入主链S中,此时在M中搜索f_i^μ所依赖的monitor,对于每一个这样的monitor,比如k∈M,有一个目前以k为终点的分支链,然后将k指向f_i^μ,从而将分支链扩展到以f_i^μ为终点,并从M中移除k;

[0079] 23)调用路径搜索算法找到所有可以并行执行的路径集合PATH;

[0080] 24)返回分支链集合B、主链S以及路径集合PATH。

[0081] 进一步的,步骤(3)所述的VNF拓扑顺序算法的目的是找到一个关于多条SFC的VNF拓扑顺序,按照该顺序部署VNF能够降低时延,具体算法流程如下:

[0082] 31)初始化f为源节点;

[0083] 32)遍历同一时刻到达的请求集合R,调用算法1得到分支链集合B和主链S以及路径集合PATH,并根据路径集合PATH求得所有路径中时延最大的路径,将其加入到集合C中;

[0084] 33)遍历C,创建有向加权图graph=(F,ω),F为VNFs的集合;

[0085] 34)调用最小反馈弧集算法求graph的最小反馈弧集,并验证所求拓扑顺序是否满足不同链间VNF的依赖关系,如果满足则返回拓扑顺序,否则返回False(即违反了依赖条件,不能使用该算法)。

[0086] 进一步的,步骤(4)所述的DPU处理策略具体描述如下:

[0087] 所述的DPU处理策略目的是接手CPU不擅长的网络协议处理、数据加解密、数据压缩等数据处理任务,以节省CPU计算资源并减少时延。

[0088] 如果某一时刻到达的请求集中有高优先级请求,判断该请求中是否有负责网络协

议处理、数据加解密、数据压缩等数据处理任务的VNF，若有则使用DPU进行快速处理，若没有，则使用CPU进行处理。

[0089] 进一步的，步骤(5)所述的基于深度强化学习的SFC异构部署算法的目的是针对不同时刻到达请求的数量和情况分别采用不同的策略进行处理，即并行策略、VNF拓扑顺序策略和DPU处理策略，以便更好地完成对SFC的部署。具体算法流程如下：

[0090] 51)系统先删除超时请求，经过优先级判断器将到达的请求R按照实时性进行划分，实时性高的划分为高优先级R_high，实时性低的划分为低优先级R_low；

[0091] 52)初始化时隙τ；

[0092] 53)根据R_high和R_low的数量决定采用何种策略对SFC进行处理；

[0093] 54)构建神经网络模型并训练，将当前物理网络的状态和正在处理的请求的特征以及上述信息作为输入，通过神经网络的计算，输出每个VNF的部署策略；

[0094] 55)更新网络状态。

[0095] 有益效果：与现有技术相比，本发明所述方法用DPU解决当前传统计算架构和传统SFC部署系统，通过CPU和DPU结合的方式解放了CPU不擅长的网络协议处理、数据加解密、数据压缩等数据处理任务，节约了计算资源，优化了处理效率。本发明基于该方法的实施包括提供一种VNF拓扑顺序算法、一种DPU处理策略和一种SFC并行策略，为设计、协调、部署和规范各种移动服务以支持日益复杂和多变的服务请求提供了一种新颖的方式，使SFC的部署更加灵活和敏捷，并且考虑到服务需求的多样性，提高服务转发性能。

附图说明

[0096] 图1为本发明所述方法的流程示意图；

[0097] 图2为本发明所述方法的系统架构图；

[0098] 图3为本发明所述VNF的依赖关系图；

[0099] 图4为本发明所述复制和合并功能图；

[0100] 图5为本发明所述SFC部署算法的流程图；

[0101] 图6为不同请求下的每个batch的平均时延对比图；

[0102] 图7为不同到达率下的每个batch的平均时延对比图；

[0103] 图8为不同请求下的请求接受率对比图；

[0104] 图9为不同到达率下的请求接受率对比图；

[0105] 图10为不同请求下的每条SFC的平均成本对比图；

[0106] 图11为不同到达率下的每条SFC的平均成本对比图；

[0107] 图12为不同请求下的每条SFC的平均reward对比图；

[0108] 图13为不同到达率下的每条SFC的平均reward对比图；

[0109] 图14为不同请求下的DPU使用率图；

[0110] 图15为不同到达率下的DPU使用率图。

具体实施方式

[0111] 为了详细的说明本发明所公开的技术方案，下面结合附图和实施例对本发明做更进一步的说明。

[0112] 本发明所提供的是一种基于CPU+DPU架构的多目标SFC并行化和部署方法，主要用于解决当前传统计算架构和传统SFC部署系统面临的转发性能下降、场景和需求多样化等问题。

[0113] 云计算、数据中心、智算中心等基础设施快速扩容，网络带宽从主流的10G朝着25G、40G、100G、200G甚至400G发展，传统基于CPU完成数据转发的模式出现瓶颈。为了提升CPU数据转发性能，Intel推出了数据平面开发工具包(DPDK)加速方案，通过绕过内核协议栈，以用户态绑核轮询的模式提升了I/O数据处理性能，极大提升了包转发速率，然而在大带宽趋势下，这种方案的CPU开销难以忽视，在25G带宽速率下，大部分数据中心仅为满足业务的数据I/O需求所需CPU开销就达到了25％甚至更多。此外，随着人工智能场景的普及，越来越多的云上AI计算任务对网络和存储I/O的时延性能提出了更极致的需求，全称远程直接数据存取(RDMA)和非易失性内存主机控制器接口规范(NVMe)等高性能网络和存储协议在传统网卡架构下难以满足云计算多租户的需求场景。这种背景下，为解决后摩尔时代I/O性能瓶颈和虚拟化技术发展限制等诸多问题，DPU应运而生。

[0114] DPU是5G时代集网络加速为一体的新型数据处理单元。DPU本质上是分类计算，是将数据处理/预处理从CPU卸载，同时将算力分布在更靠近数据发生的地方，从而降低通信量，其内部融合了RDMA、网络功能、存储功能、安全功能、虚拟化功能。它可以用来接手CPU不擅长的网络协议处理、数据加解密、数据压缩等数据处理任务，同时兼顾传输和计算的需求。此外，DPU还起到连接枢纽的作用，一端连接CPU、GPU、固态硬盘(SSD)、FPGA加速卡等本地资源，一端连接交换机/路由器等网络资源。总体而言，DPU不仅提高了网络传输效率，而且释放了CPU算力资源，从而带动整体数据中心的降本增效。相比于传统网卡，DPU在单部件成本上有所增加，但是DPU的引入解放了更高成本的主机CPU算力，释放了更多可售卖资源，因此DPU引发的架构变革一定程度上提升了整个数据中心资源池的能效成本比和公有云厂商的收益成本比。未来DPU可以覆盖越来越多的需求和场景。同时由于NFV的引入使得SFC的部署面临着转发效率低、服务需求多样化等问题。在这一机遇背景下，本发明研究由CPU和DPU组成的异构计算架构，并将该架构应用在解决SFC部署问题中。

[0115] 下面具体说明本发明所提供的技术方案实施过程。

[0116] 本发明所述方法是以CPU+DPU为计算架构实现对SFC的部署。主要包括编排器和服务器。编排器负责接收来自网络运营商的SFC请求，运行SFC部署算法以确定哪些SFC被接受以及这些被接受的SFC如何被放置。由异构计算结构组成的服务器负责根据编排器传达的部署策略分别使用CPU或DPU完成对SFC的部署。

[0117] 本发明所述方法主要实施流程如附图1所示，基于上述的技术方案，在实施例中做进一步的详细说明，具体包括如下步骤：

[0118] (1)构建一个异构计算架构来解决多目标部署问题，该架构包括一个负责管理和协调的编排器和基于CPU+DPU结构的服务器；

[0119] (2)依据VNF间的独立性，提出SFC并行问题，并提出一种SFC并行策略将串行SFC转化为并行SFC；

[0120] (3)针对某一时刻多条SFC同时到达的情况，提出一种VNF拓扑顺序算法，按照该算法得到的拓扑顺序部署VNF，并结合VNF的共享和缩放特性能够显著降低时延；

[0121] (4)针对实时性要求高的服务请求，提出一种DPU处理策略，即根据请求到达的情

况和数量使用DPU对其进行快速处理；

[0122] (5)提出基于深度强化学习的SFC异构部署算法，该异构算法能够针对不同时刻到达请求的数量和情况分别完成对SFC的部署。

[0123] 下面具体阐述其实施过程。

[0124] 1、构建异构计算架构

[0125] 如附图2所示，异构计算架构由编排器和服务器组成，其中编排器包括SFC部署算法模块，资源管理模块以及驱动模块；服务器包括CPU、DPU以及连接CPU和DPU的总线。编排器负责管理和部署到达的SFC，而由异构计算结构组成的服务器则负责根据编排器传达的部署策略依次处理不同SFC中的VNF。其中编排器的具体任务包括接收来自网络运营商的SFC请求，运行SFC部署算法以确定哪些SFC被接受以及这些被接受的SFC如何被放置。为了考虑不同请求的不同情况，该算法将分别调用并行策略、VNF拓扑顺序策略和DPU处理策略，以获得每个请求的最佳部署方案，然后调用资源管理模块来管理资源，最后调用驱动模块将部署方案传递给放置的服务器，由服务器根据部署方案分别使用CPU或DPU完成对SFC的部署。

[0126] 2、结合实际以及所提出的计算架构提出多目标SFC部署问题

[0127] SFC部署问题的目标函数如下：

[0128] $\min(f_1+f_2+f_3)$

[0129] $s.t.C_1,C_2,C_3$

[0130] $$\begin{cases} f_1=\sum_{r_\mu\in R} D_\mu \\ f_2=\sum_{r_\mu\in R} y_{r_\mu} B_\mu \tau_r \\ f_3=C(\tau) \end{cases}$$

[0131] 对于f_1，D_μ为总体响应时延，即

[0132] $$D_\mu=L_\mu+P_\mu+T_\mu+\overline{W_q}$$

[0133] 其中$L_\mu=\sum_{e_h^\mu\in E_\mu}\sum_{e_j\in E} x_{e_h^\mu}^{e_j} D_{e_j}$为通信时延，$P_\mu=\sum_{f_v^\mu\in F_\mu}\sum_{n_i\in N} x_{f_v^\mu}^{n_i}\cdot\frac{1}{\frac{\eta_{m_i}^\mu c_{m_i}}{w_{m_i}^\mu}-\lambda_\mu+\varepsilon}$为处理时延，$T_\mu=\sum_{f_v^\mu\in F_\mu}\frac{U}{v_\mu}$为传输时延，$\overline{W_q}$为平均排队时延。

[0134] 对于f_2，$\sum_{r_\mu\in R} y_{r_\mu} B_\mu \tau_r$为总吞吐量，其中二进制变量$y_{r_\mu}$表示$r_\mu$是否被接受，$B_\mu$为SFC的最小带宽，$\tau_r=1*\Delta$表示SFC的生存时间。

[0135] 对于f_3，$C(\tau)$表示总的部署成本，即

[0136] $C(\tau)=SC(\tau)+C_{scale}(\tau)+C_{DPU}(\tau)$

[0137] $SC(\tau)$表示总运营成本，即为服务器打开成本和成功放置VNF的成本之和：

[0138] $$SC(\tau)=\sum_{n_i\in N}\sum_{f_v^\mu\in F_\mu} x_{f_v^\mu}^{n_i}\zeta_c C_{f_v^\mu}+\sum_{e_j\in E}\sum_{e_h^\mu\in E_\mu} x_{e_h^\mu}^{e_j}\zeta_B B_\mu+\sum_{n_i\in N}\zeta_O$$

[0139] $x_{f_v^\mu}^{n_i}$表示请求$r_\mu\in R$中，VNF $f_v^\mu\in F_\mu$是否部署在服务器节点$n_i\in N$上，$x_{e_h^\mu}^{e_j}$表示请求$r_\mu\in$

R中，虚拟链路$e_h^{\mu} \in E_{\mu}$是否映射到物理链路$e_j \in E$上，ζ_c和ζ_B分别表示资源和带宽的单位成本，$C_{f_v^{\mu}}$表示VNF $f_v^{\mu} \in F_{\mu}$的资源需求，ζ_0表示服务器打开成本。

[0140] $C_{scale}(\tau) = \sum_{n_i \in N} \sum_{f_v^{\mu} \in F_{\mu}} C^h \cdot x_{n_i,h}^{f_v^{\mu}}$表示总缩放成本，其中$C^h$表示水平缩放的成本，$x_{n_i,h}^{f_v^{\mu}}$表示VNF $f_v^{\mu} \in F_{\mu}$是否进行水平缩放。

[0141] $C_{DPU}(\tau)$表示DPU的总使用成本，定义如下：

[0142] $$C_{DPU}(\tau) = \sum_{n_i \in N} \sum_{f_v^{\mu} \in F_{\mu}} \zeta_{c_D} x_{n_i,D}^{f_v^{\mu}} C_{f_v^{\mu}} + \zeta_{B_D} x_{n_i,D}^{f_v^{\mu}} B_{\mu}$$

[0143] 其中，ζ_{c_D}和ζ_{B_D}使用DPU时资源和带宽的单位成本，$x_{n_i,D}^{f_v^{\mu}}$表示VNF $f_v^{\mu} \in F_{\mu}$是否使用DPU进行处理。

[0144] 资源约束如下：

[0145] $$C_1: \ \forall n_i \in N, \sum_{f_v^{\mu} \in F_{\mu}} \sum_{l \in N^{f_v^{\mu}}} s_{n_i,\tau}^{f_v^{\mu}} \cdot C_{f_v^{\mu}} \leq C_{n_i}$$

[0146] 其中，$s_{n_i,\tau}^{f_v^{\mu}}$表示部署在节点$n_i \in N$上的VNF $f_v^{\mu} \in F_{\mu}$的服务实例的数量，C_{n_i}表示节点$n_i \in N$的资源(cpu和memory)大小。

[0147] 带宽约束如下：

[0148] $$C_2: \ \forall e_j \in E, \sum_{r_{\mu} \in R} \sum_{e_h^{\mu} \in E_{\mu}} x_{e_h^{\mu}}^{e_j} \cdot a_{r,\tau} \cdot B_{\mu} \leq B_{e_j}$$

[0149] 其中，$a_{r,\tau}$表示请求$r_{\mu} \in R$是否仍在服务当中，B_{e_j}表示节点$n_i \in N$的带宽大小。

[0150] 延迟约束如下：

[0151] $$C_3: \ \forall r_{\mu} \in R, D_{\mu} \leq D_{\mu}^{max}$$

[0152] 其中，D_{μ}^{max}表示最大最大端到端延迟大小。

[0153] 3、结合实际提出SFC并行问题

[0154] 在SFC中，某些VNF可以独立工作而不影响其他VNF，所以可以将串行SFC转换为并行SFC。然而，并不是SFC中的所有VNF都可以并行工作，如果当两个VNF都修改流的内容或违反依赖性约束时，两个VNF的操作是冲突的。只有当VNF在SFC中相互独立时，才能在VNF之间并行处理；否则，网络和服务策略的正确性可能会被破坏。

[0155] 本发明将VNF分成两种类型：monitors和shapers，其中monitors负责监测流量而不作任何修改，而shapers用来处理和修改流量。由于SFC中的VNF必须按照特定的顺序应用于每个数据包流，因此它们形成了一种依赖关系。本发明中规定，如果一个VNF f_v^{μ}在另一个VNF f_{v+1}^{μ}之前，称f_{v+1}^{μ}依赖于f_v^{μ}，记为$f_v^{\mu} \prec f_{v+1}^{\mu}$。不同VNF之间的依赖关系见附图3。

[0156] 为了并行地处理数据包，需要两个功能。1)复制功能和2)合并功能。如附图4所示，当一个数据包进来时，复制功能会复制该数据包并将其发送给可以并行处理的VNF。在数据包被处理后，复制的数据包被合并功能合并。数据包复制和合并会引起额外的资源消耗和

时延，因此在SFC并行问题中，目标函数是最小化由数据包复制和合并引起的额外的资源消耗和时延：

[0157] $\min(\alpha C+\beta\Delta d)$

[0158] 其中α和β分别表示额外的资源消耗和时延的权重系数。

[0159] C表示数据包复制和合并引起的额外资源消耗，公式如下：

[0160] $$C=\sum_{\mu=1}^{|R|}\sum_{\forall B\in r_\mu}64\times\frac{\Phi_B-1}{U}$$

[0161] 其中B是一组并行分支，Φ_B表示B的并行度，U表示数据包的大小。

[0162] 数据包复制和合并引起的额外延迟Δd可表示为：

[0163] $$\forall B\in r_\mu\in R,\Delta d=\sum_{\mu=1}^{|R|}30\times\frac{V_\mu}{10^9}$$

[0164] 其中V_μ是请求R中第μ个SFC的数据量。

[0165] 对于SFC并行问题，引入流量约束。用O_c表示r_μ中复制节点的集合，O_m表示r_μ中合并节点的集合。$O_c(n_i)$和$O_m(n_i)$分别表示r_μ中复制节点和合并节点的数量。对于SFC r_μ，除了复制节点、合并节点、源节点和目的节点不满足流量守恒外，所有中间节点都满足流量守恒，即：

[0166] $$\forall r_\mu\in R,\forall n_h\in N,n_h\notin\{n_{src},n_{dst},O_c,O_m\}:$$

[0167] $$\sum_{e_h^\mu\in E_\mu}\sum_{n_g\in N}x_{e_h^\mu}^{n_g,n_h}-\sum_{e_h^\mu\in E_\mu}\sum_{n_k\in N}x_{e_h^\mu}^{n_h,n_k}=0$$

[0168] 如果源节点是一个复制节点，则需要满足该约束条件：

[0169] $$\forall r_\mu\in R,\forall n_{src}\in Oc:$$

[0170] $$\sum_{e_h^\mu\in E_\mu}\sum_{n_u\in N}x_{e_h^\mu}^{n_{src},n_u}-\sum_{e_h^\mu\in E_\mu}\sum_{n_v\in N}x_{e_h^\mu}^{n_v,n_{src}}=O_c(n_{src})$$

[0171] 如果目的节点是一个合并节点，则需要满足如下约束条件：

[0172] $$\forall r_\mu\in R,\forall n_{dst}\in Om:$$

[0173] $$\sum_{e_h^\mu\in E_\mu}\sum_{n_v\in N}x_{e_h^\mu}^{n_v,n_{dst}}-\sum_{e_h^\mu\in E_\mu}\sum_{n_u\in N}x_{e_h^\mu}^{n_{dst},n_u}=O_m(n_{dst})$$

[0174] 对于其他节点是复制节点的情况，应该满足以下公式：

[0175] $$\forall r_\mu\in R,\forall n_h\in Oc,\sum_{f_v^\mu\in F_\mu}x_{f_v^\mu}^{n_h}=1:$$

[0176] $$\sum_{e_h^\mu\in E_\mu}\sum_{n_u\in N}x_{e_h^\mu}^{n_h,n_u}-\sum_{e_h^\mu\in E_\mu}\sum_{n_v\in N}x_{e_h^\mu}^{n_v,n_h}=O_c(n_h)-O_m(n_h)$$

[0177] 对于其他节点是合并节点的情况，需要满足以下公式：

[0178] $$\forall r_\mu\in R,\forall n_h\in Om,\sum_{f_v^\mu\in F_\mu}x_{f_v^\mu}^{n_h}=1:$$

[0179] $$\sum_{e_h^\mu\in E_\mu}\sum_{n_v\in N}x_{e_h^\mu}^{n_v,n_h}-\sum_{e_h^\mu\in E_\mu}\sum_{n_u\in N}x_{e_h^\mu}^{n_h,n_u}=O_m(n_h)-O_c(n_h)$$

[0180] 4、设计SFC并行策略

[0181] 所述SFC并行策略目的是根据VNF之间的依赖关系识别链中的VNF以找到所有可以并行执行的链。所述SFC并行策略具体算法流程如下：

[0182] 1)初始化分支链集合B、主链S，monitor集合M；

[0183] 2)遍历r_μ，如果$f_i^\mu \in r_\mu$为monitor，首先初始化一个分支链b∈B，然后将f_i^μ加入b以及M中；如果$f_i^\mu \in r_\mu$为shaper，则将其加入主链S中，此时在M中搜索f_i^μ所依赖的monitor，对于每一个这样的monitor，比如k∈M，有一个目前以k为终点的分支链，然后将k指向f_i^μ，从而将分支链扩展到以f_i^μ为终点，并从M中移除k；

[0184] 3)调用路径搜索算法找到所有可以并行执行的路径集合PATH；

[0185] 4)返回分支链集合B、主链S以及路径集合PATH。

[0186] 5、设计VNF拓扑顺序策略

[0187] 所述VNF拓扑顺序策略的目的是找到一个关于多条SFC的VNF拓扑顺序，按照该顺序部署VNF能够降低时延，具体算法流程如下：

[0188] 1)初始化f为源节点；

[0189] 2)遍历同一时刻到达的请求集合R，调用算法1得到分支链集合B和主链S以及路径集合PATH，并根据路径集合PATH求得所有路径中时延最大的路径，将其加入到集合C中；

[0190] 3)遍历C，创建有向加权图graph＝(F,ω)，F为VNFs的集合；

[0191] 4)调用最小反馈弧集算法求graph的最小反馈弧集，并验证所求拓扑顺序是否满足不同链间VNF的依赖关系，如果满足则返回拓扑顺序，否则返回False(即违反了依赖条件，不能使用该算法)。

[0192] 6、设计DPU处理策略

[0193] 所述的DPU处理策略目的是接手CPU不擅长的网络协议处理、数据加解密、数据压缩等数据处理任务，以节省CPU计算资源并减少时延。

[0194] 如果某一时刻到达的请求集中有高优先级请求，判断该请求中是否有负责网络协议处理、数据加解密、数据压缩等数据处理任务的VNF，若有则使用DPU进行快速处理，若没有，则使用CPU进行处理。

[0195] 7、基于上述提出的策略和架构设计基于深度强化学习的SFC异构部署算法

[0196] 所述基于深度强化学习的SFC异构部署算法的目的是针对不同时刻到达请求的数量和情况分别采用不同的策略进行处理，即并行策略、VNF拓扑顺序策略和DPU处理策略，以便更好地完成对SFC的部署。具体算法流程如下：

[0197] 1)系统先删除超时请求，经过优先级判断器将到达的请求R按照实时性进行划分，实时性高的划分为高优先级R_high，实时性低的划分为低优先级R_low；

[0198] 2)初始化时隙τ；

[0199] 3)根据R_high和R_low的数量决定采用何种策略对SFC进行处理，详见附图5；

[0200] 4)构建神经网络模型并训练，将当前物理网络的状态和正在处理的请求的特征以及上述信息作为输入，通过神经网络的计算，输出每个VNF的部署策略；

[0201] 5)更新网络状态。

[0202] 8、根据部署策略完成对SFC的部署

[0203] 异构计算架构中的编排器调用驱动模块将部署方案传递给放置的服务器，由服务器根据部署方案分别使用CPU或DPU完成对SFC的最佳部署，以到达降低时延和成本的效果。

[0204] 本实施例中为了验证本发明(PSHD)的实际效果，在以请求数量为控制变量的情况下和其他两个算法(BASE和FCPM)进行了模拟对比实验。由于BASE和PSHD的目标是相同的，因此本实施例中还进行了一组以请求到达率为控制变量的实验，以证明本发明的有效性。

[0205] 图6比较了服务节点数量为12，请求数量从50-300变化时，三个算法每个batch的平均时延。由图可知，本发明所述方法的时延总是最小的，随着请求数量的增加，本发明所述方法和其他算法时延的差距呈增大趋势，这说明请求数量越多，本发明所述方法降低时延的性能越好，分别比BASE、FCPM的时延低37.73％和34.26％。图7也说明在请求数量为100，请求的到达率从0.5-3变化时，本发明所述方法的时延总是最低的。随着请求到达率的增大，可以看到本发明所述方法和BASE时延的差距越来越大，这说明同一时刻到达的请求数量越多，本发明所述方法降低时延的性能越好，比BASE时延低47.04％。此外，图6和图7中PD为不采用VNF Topological Order策略时的时延走向，由图可知PD和本发明所述方法时延差距逐渐增大，这证明VNF Topological Order策略降低时延的有效性。

[0206] 本实施例中探讨了与最先进的方法相比，本发明所述方法的请求接受率。图8描述了服务节点数量为12，请求数量从50-300变化时的结果。图9描绘了请求数量为100，请求的到达率从0.5-3变化时的结果。由图8所示，本发明所述方法的请求接受率是最高的，平均为0.728，其次是BASE算法，平均为0.668，FCPM的接受率最低，为0.517。图9同样证明本发明所述方法的请求接受率总是最高的，平均为0.719。由图可知，随着到达率的增大，BASE的请求接受率下降的趋势明显大于本发明所述方法，证明某一时刻到达的请求越多，BASE的部署能力越差，本发明所述方法的部署能力越强。

[0207] 本实施例中比较了服务节点数为12，请求数从50-300变化时，三个算法每条SFC的平均成本。如图10所示，本发明所述方法的成本最高，并且随着请求数量的增加而增多，这是因为本发明所述方法引入了CPU+DPU的异构架构，即会根据情况使用DPU进行快速处理，而DPU的使用成本较高一些。图11也说明在请求数量为100，请求的到达率从0.5-3变化时，本发明所述方法的成本总是最高的，平均为8.61，而BASE的成本平均为4.55。本发明所述方法成本的增加带来的是请求接受率的提高以及时延的降低，如图6-图9所示，并且由图12-13所示，本发明所述方法的reward总是最高的。以上证明，牺牲部分成本能够带来更好的收益。

[0208] 最后，本实施例中比较了本发明所述方法在服务节点数量为12，请求数量从50-300变化以及请求数量为100，请求的到达率从0.5-3变化时的DPU使用率。如图14和图15所示，DPU的使用率随请求数量/请求到达率的增加而增大，DPU使用率的增加带来的是成本的增加，如图10-11，相较于其他算法而言较高的请求接受率，如图8-9，以及较低的时延，如图6-7，结合图12三个算法的reward比较可知，本发明所述方法的性能远高于其他算法。

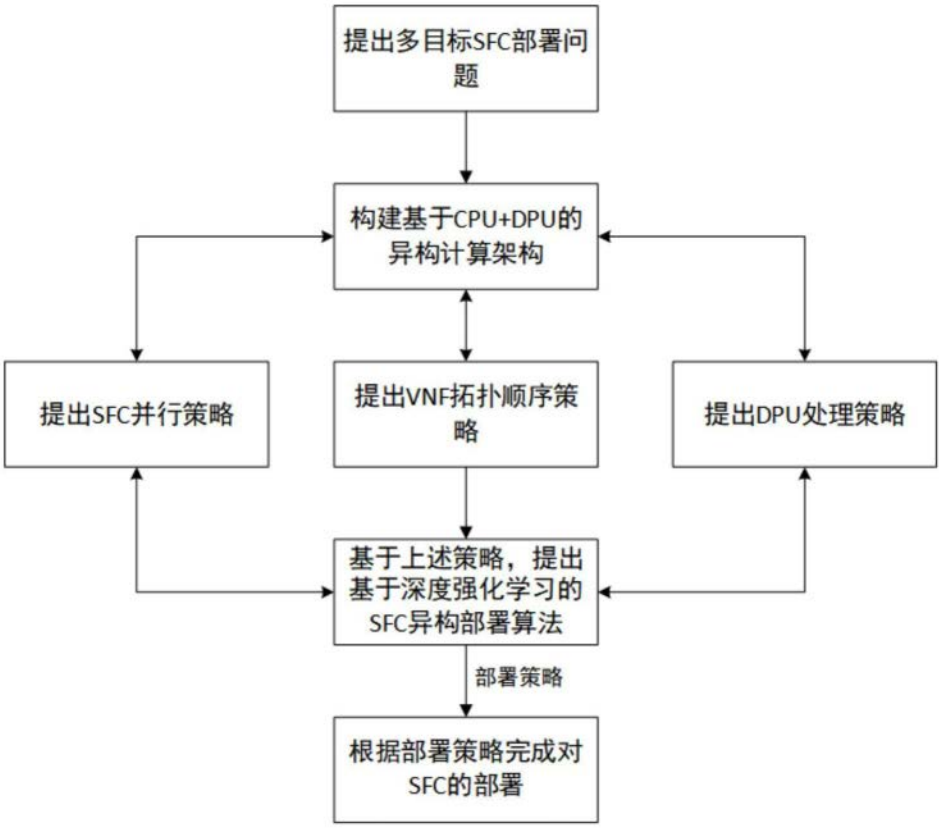

提出多目标SFC部署问题
构建基于CPU+DPU的异构计算架构
提出SFC并行策略
提出VNF拓扑顺序策略
提出DPU处理策略
基于上述策略，提出基于深度强化学习的SFC异构部署算法
部署策略
根据部署策略完成对SFC的部署

图1

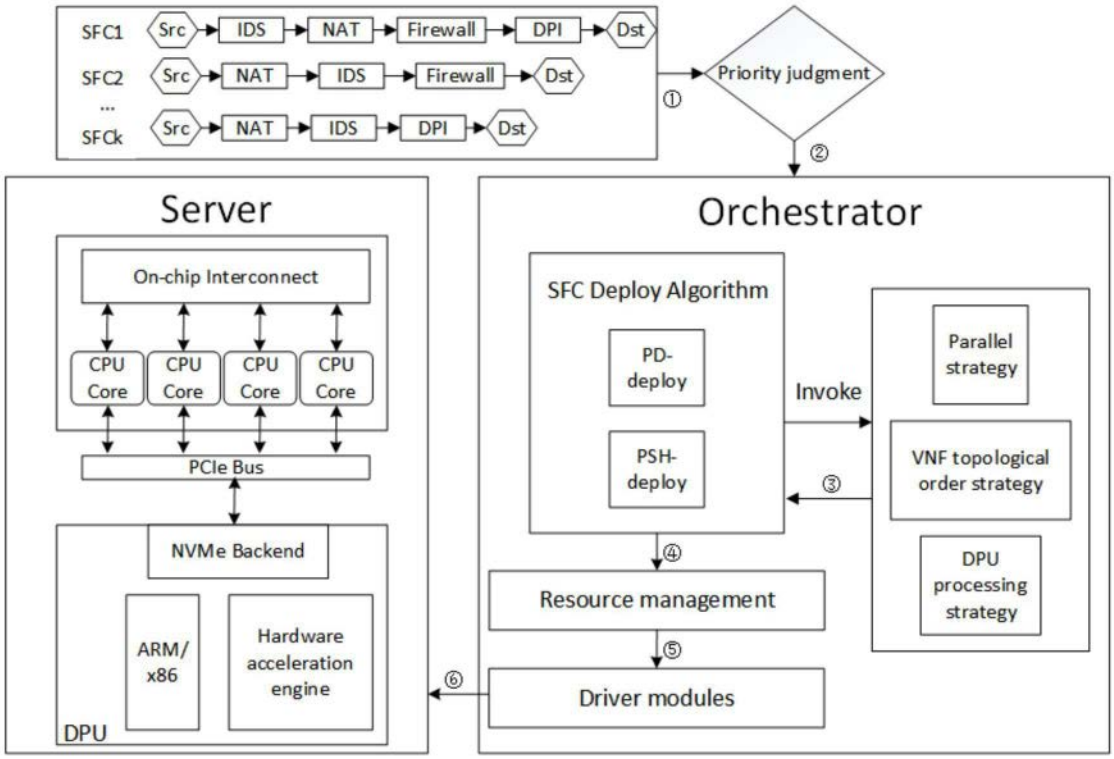

SFC1
SFC2
...
SFCk
Src
IDS
NAT
Firewall
DPI
Dst
①
Priority judgment
②
Server
On-chip Interconnect
CPU Core
CPU Core
CPU Core
CPU Core
PCIe Bus
NVMe Backend
ARM/ x86
Hardware acceleration engine
DPU
⑥
Orchestrator
SFC Deploy Algorithm
PD-deploy
PSH-deploy
Invoke
③
④
Resource management
⑤
Driver modules
Parallel strategy
VNF topological order strategy
DPU processing strategy

图2

Type	VNF	Dependent relationship	VNF features
Monitor	Traffic Logger (TL)	$succprec-shapers$	Record packets and their statistics
Monitor	Traffic Visualizer (TV)	$succprec-shapers$	Visualize the statistics of traffic flows
Monitor	Packet Header Inspector(PHI)	$succprec-shapers, precsucc-shapers$	Check packet headers and decide to forward or drop any packet
Monitor	Deep Packet Inspector(DPI)	$succprec-shapers, precsucc-shapers$	Check packet contents and decide to forward or drop any packet
Monitor	DDoS Scrubber (DS)	$succprec-shapers, precsucc-shapers$	Identify and remove malicious traffic flows
Shaper	Network Address Translator (NAT)	$succprec-shapers, PHI, DPI, DS, precsucc-shapers$	Modify IP header to map one IP address space into another
Shaper	Traffic Zipper (TZ)	$succprec-shapers, PHI, DPI, DS, precsucc-shapers, TU$	Compress packets to shrink the traffic volume
Shaper	Traffic Unzipper (TU)	$succprec-shapers, PHI, DPI, DS, precsucc-shapers$	Uncompress the compressed packets into the original format
Shaper	Traffic Encryptor (TE)	$succprec-shapers, PHI, DPI, DS, precsucc-shapers, TD$	Convert packets into cipher text
Shaper	Traffic Decryptor (TD)	$succprec-shapers, PHI, DPI, DS, precsucc-shapers$	Recover the cipher text into plain text

图3

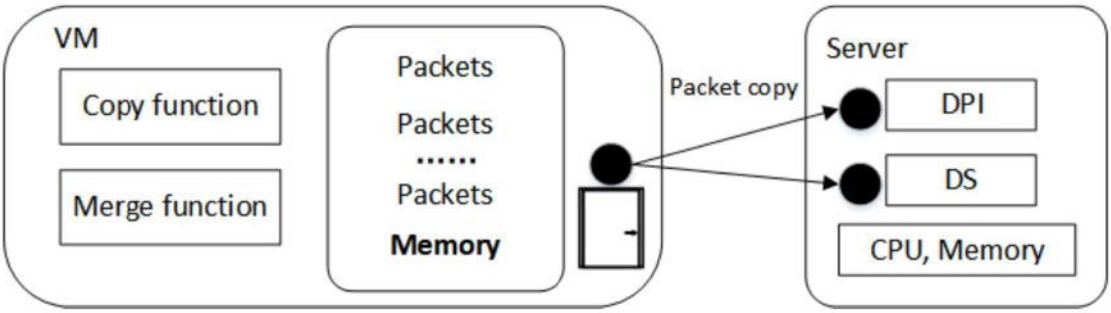


图4

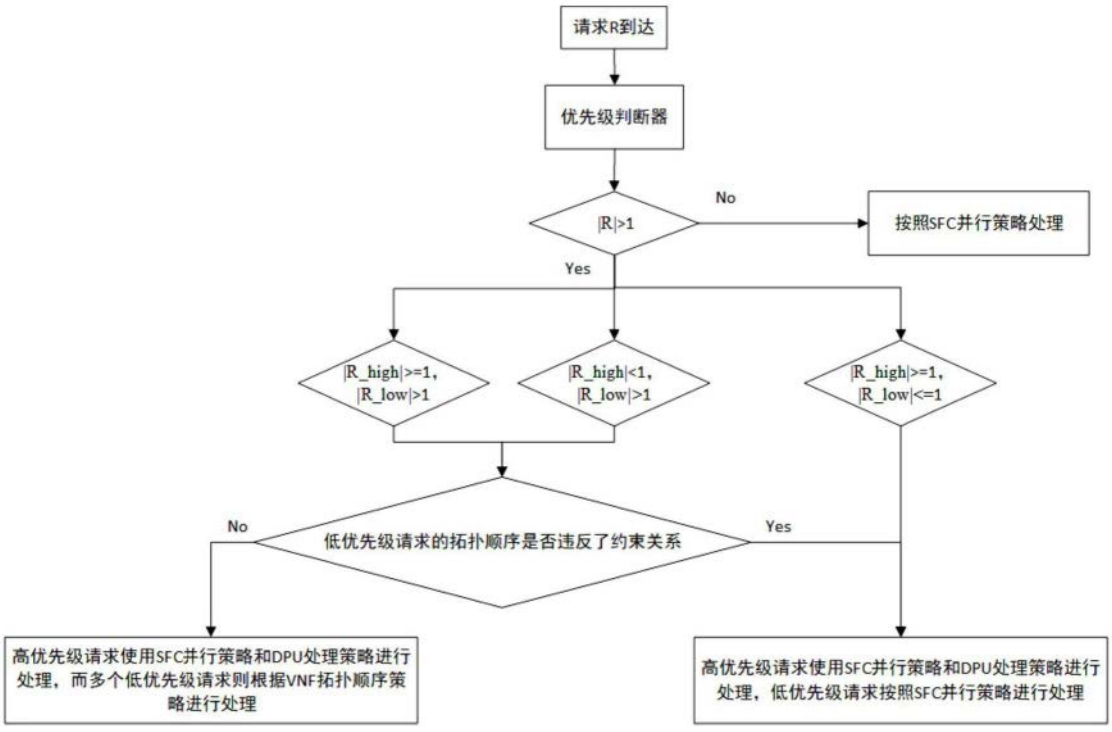


图5

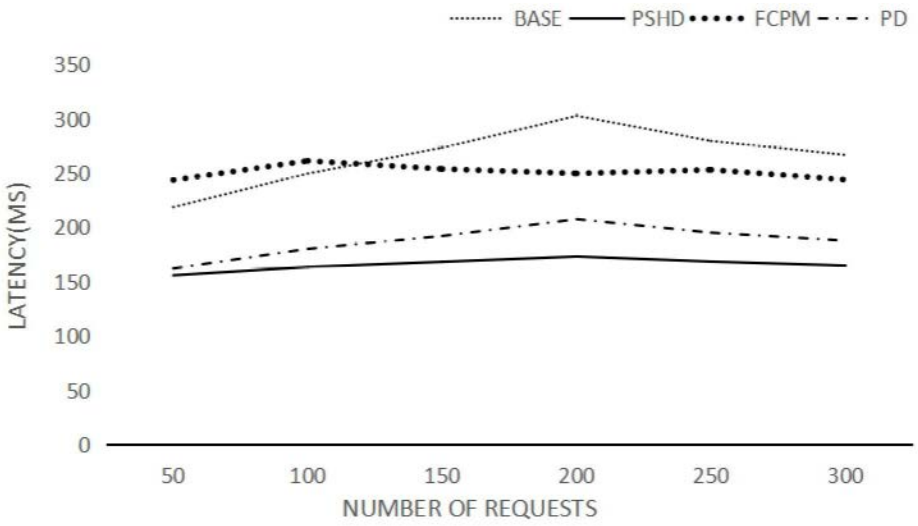

BASE
PSHD
FCPM
PD
350
300
250
200
150
100
50
0
LATENCY(MS)
50
100
150
200
250
300
NUMBER OF REQUESTS

图6

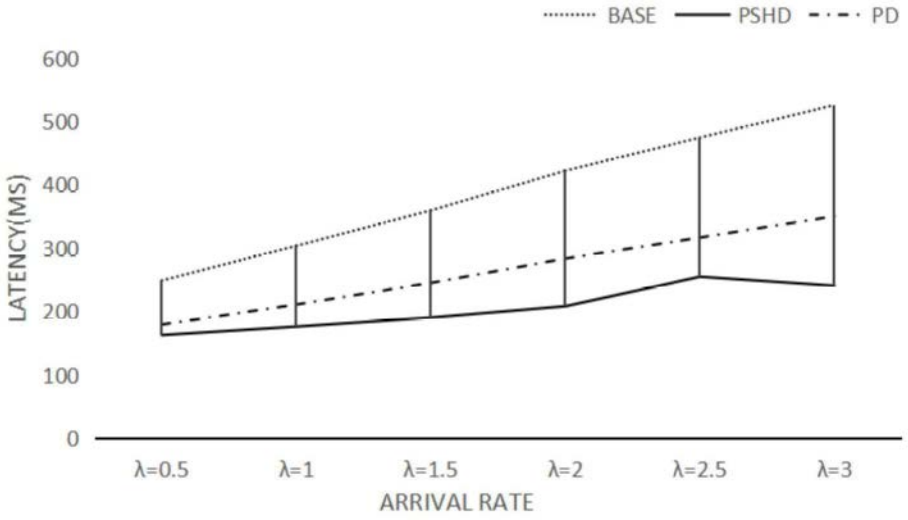

BASE
PSHD
PD
600
500
400
300
200
100
0
LATENCY(MS)
λ=0.5
λ=1
λ=1.5
λ=2
λ=2.5
λ=3
ARRIVAL RATE

图7

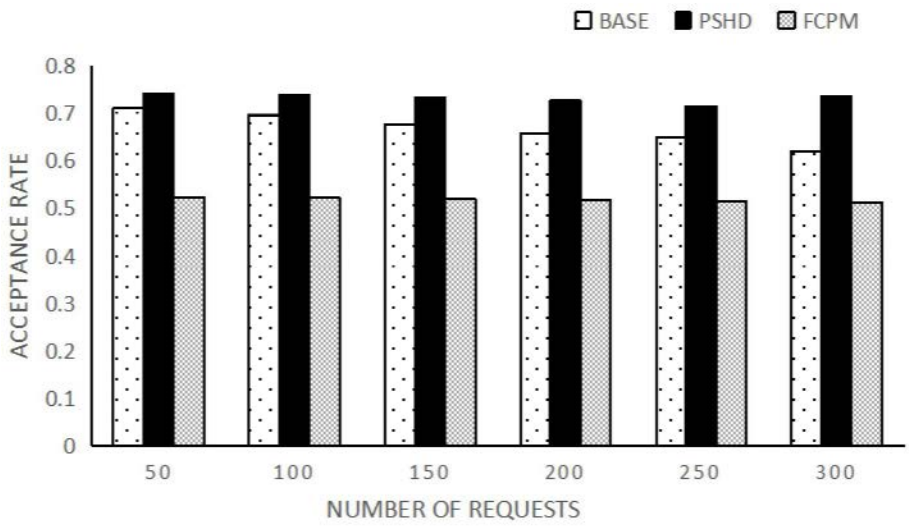

BASE
PSHD
FCPM
ACCEPTANCE RATE
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
300
NUMBER OF REQUESTS

图8

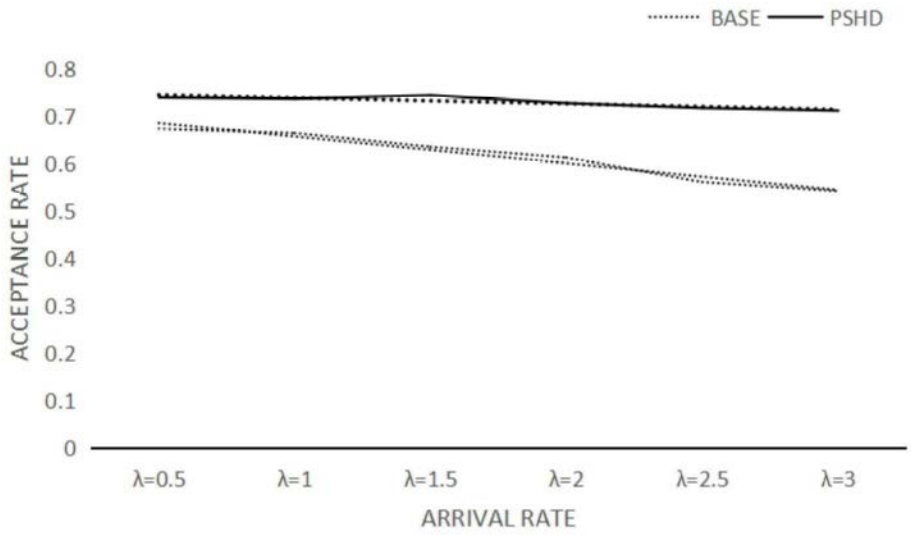

BASE
PSHD
ACCEPTANCE RATE
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
λ=0.5
λ=1
λ=1.5
λ=2
λ=2.5
λ=3
ARRIVAL RATE

图9

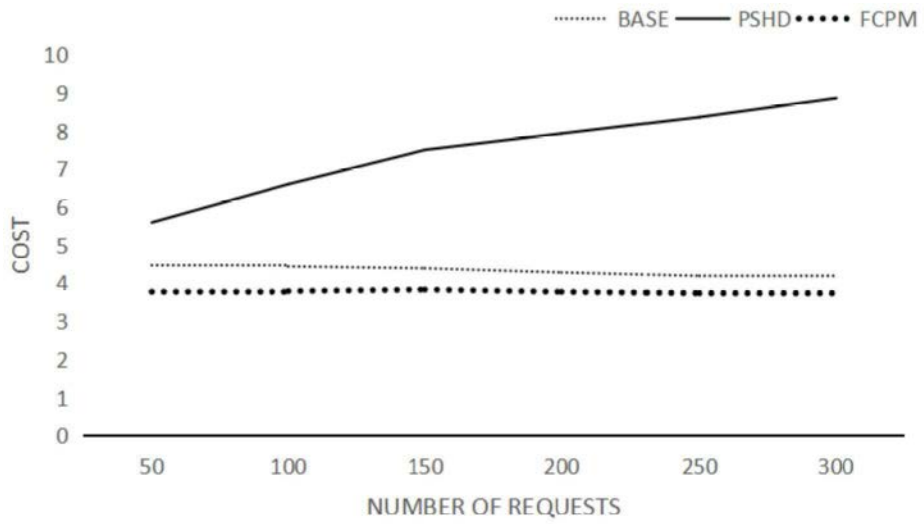

BASE
PSHD
FCPM
10
9
8
7
6
5
4
3
2
1
0
COST
50
100
150
200
250
300
NUMBER OF REQUESTS

图10

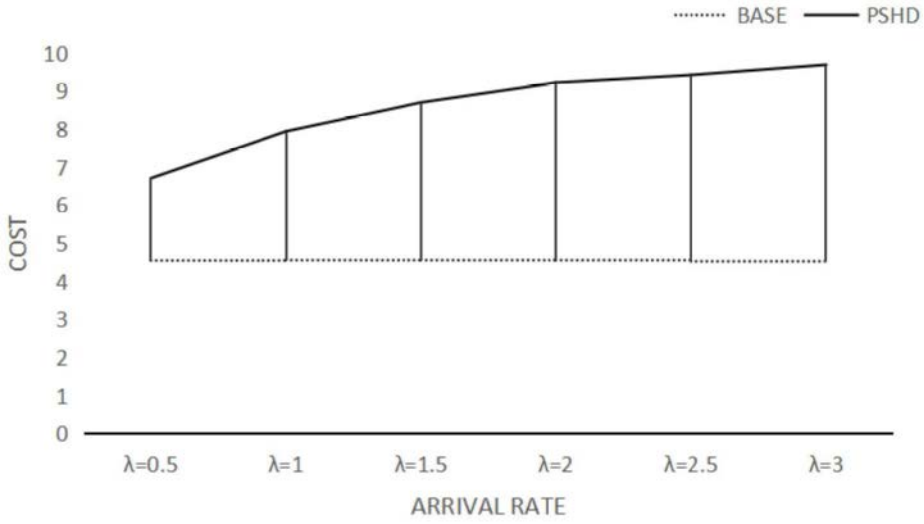

BASE
PSHD
10
9
8
7
6
5
4
3
2
1
0
COST
λ=0.5
λ=1
λ=1.5
λ=2
λ=2.5
λ=3
ARRIVAL RATE

图11

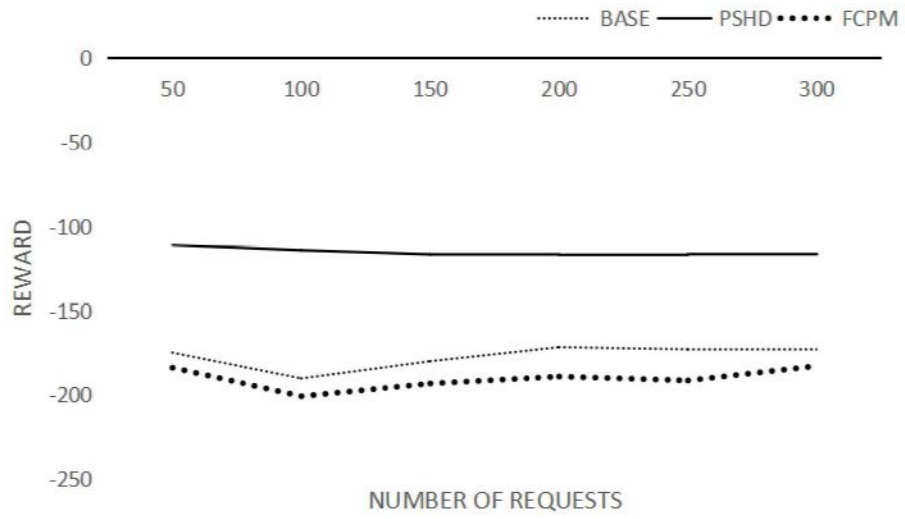

BASE
PSHD
FCPM
0
50
100
150
200
250
300
-50
-100
-150
-200
-250
REWARD
NUMBER OF REQUESTS

图12

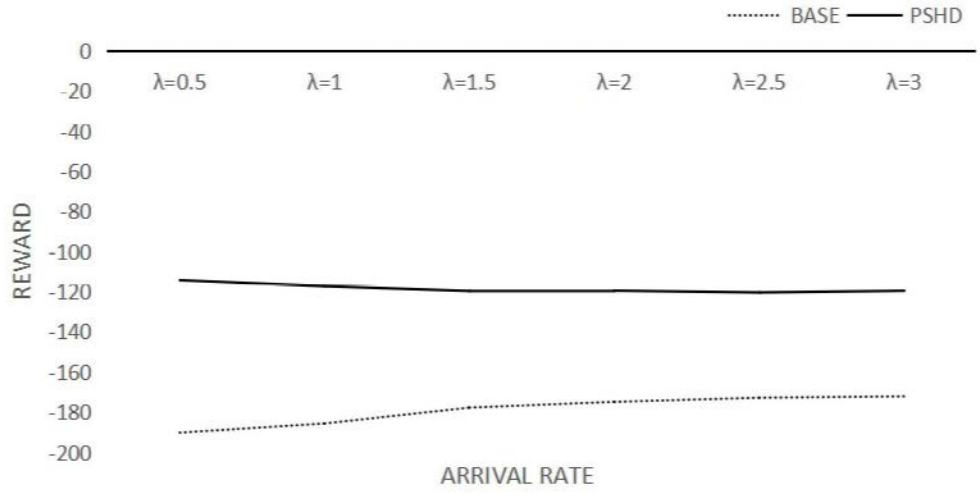

BASE
PSHD
0
λ=0.5
λ=1
λ=1.5
λ=2
λ=2.5
λ=3
-20
-40
-60
-80
-100
-120
-140
-160
-180
-200
REWARD
ARRIVAL RATE

图13

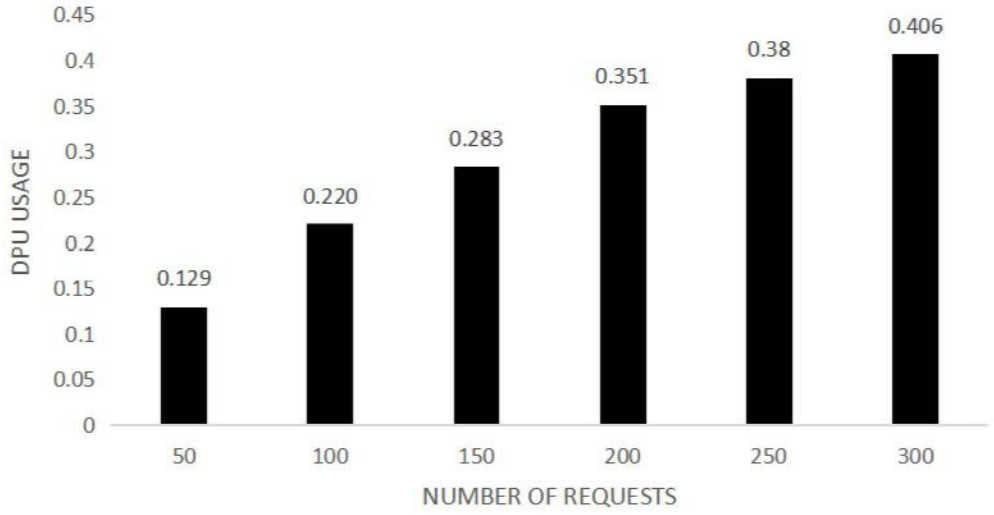

0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
DPU USAGE
0.129
0.220
0.283
0.351
0.38
0.406
50
100
150
200
250
300
NUMBER OF REQUESTS

图14

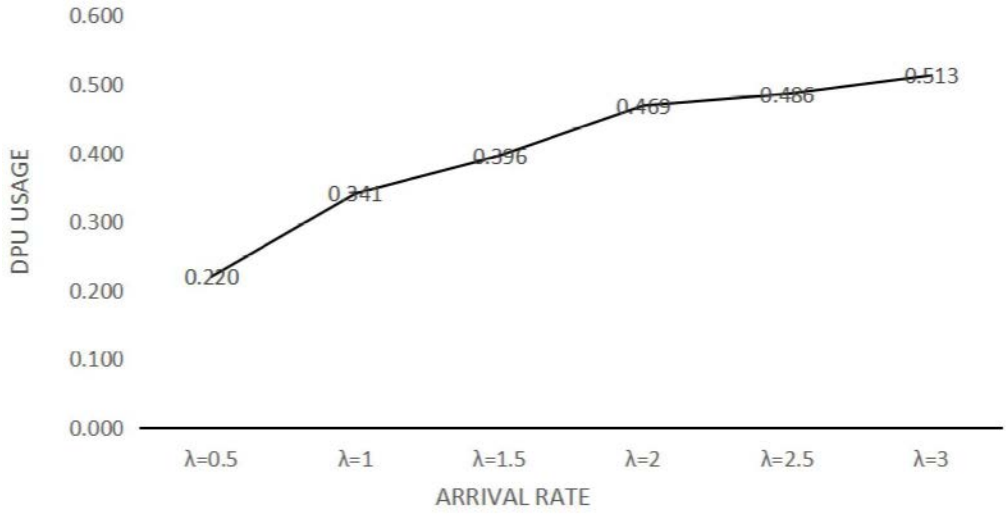

0.600
0.500
0.400
0.300
0.200
0.100
0.000
DPU USAGE
0.220
0.341
0.396
0.469
0.486
0.513
λ=0.5
λ=1
λ=1.5
λ=2
λ=2.5
λ=3
ARRIVAL RATE

图15