The Testbed of NFV

费君尧5130309739

2016 年 6 月 26 日

1 Introduction

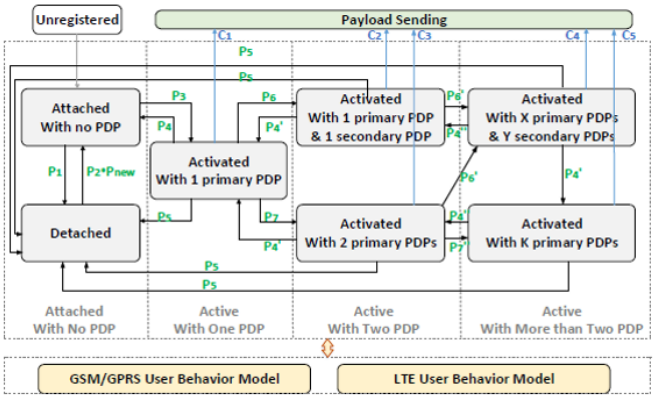
It is based on the paper "Resource Allocation for Virtualized Mobile Core Networks: A Synthetic Approach". The NFV is to virtualize the net work function with the virtual mechines to simulate a network on the server. The target is to find a better method to allocate the resources in the cloud data center according to the data.

2 Theory Support

This work is based on the user behavior model.
We model users' behaviors using a Markov chain basedon our data analysis. Our model focuses on users' behaviors when they are using Internet applications, because the challenges of operators are mostly from such services. A user communicates with the Internet through establishing packet data protocol (PDP) contexts (3G) and bearers (4G) in the evolved packet core (EPC), respectively; therefore, we are interested in modeling the user behavior in those two cases.

3 Resource Demand Estimation

VNFs are visited when the user transits from one state to another, which leads to resource consumptions at those visited NFs. This is because each state means a procedure. This section is to show the relationship between each user behavior (i.e., user state transition) and the VNFs' resource consumption. We reveal the state-VNF relationship from 3GPP communication protocols and the



State	Explanations
S_1	Detached
S_2	Attached
S_3	Activated 1 primary PDP
S_4	Activated 1 primary PDP & 1 secondary PDP
S_5	Activated 2 primary PDPs
S_6	Activated X primary PDPs & Y secondary PDPs
S_7	Activated K primary PDPs
S_8	Payload Sending
S_9	GSM & GPRS 2G network
S_{10}	LTE 4G network
S_{11}	Unregistered

Figure 1: The user behavior model

description files of real mobile systems. With such information, we can build a state-VNF matrix, where each element means the weights of the use of NFs in each process. Such weights for memory are obtained by observing the number of visits the state could pay to the NF, and that for CPU by observing the time complexity of specific operations for CPU in the entire procedure. In this work, we choose 6 kinds of most important NFs: MME, SGSN, GGSN, S-GW, P-GW, PCRF and HSS(HLR),because we focus on the mobile core networks. Assume there are m kinds of user behavior states and n kinds of network function nodes. The mapping matrix Mif should be weight matrix with m rows and n columns, where m_{ij} represents the relative load on the Nf function node.

In general, the total demand of each V NFj for each kind of resource can be

CPU of Nodes	MME	S-GW	P-GW	PCRF	HSS	SGSN
Attach	4	3	3	3	3	0
Detach	3	0	1	1	0	2
Activate	0	0	2	2	0	0
Deactivate	2	1	1	1	0	0
Secondary	0	2	3	4	0	0

Figure 2: State-NF CPU Mapping Matrix

CPU of Nodes	MME	S-GW	P-GW	PCRF	HSS	SGSN
Attach	19	11	6	3	6	0
Detach	6	4	3	1	0	2
Activate	5	4	4	2	0	0
Deactivate	4	6	4	1	0	0
Secondary	2	6	3	4	0	5

Figure 3: State-NF Memory Mapping Matrix

calculated as:

$$R(CPU)_j = \sum_i M(CPU)_{ij} * (\sum_k P_i * S_k)$$

$$R(Men)_j = \sum_i M(Men)_{ij} * (\sum_k P_i * S_k)$$

We first calculate the total number of each procedures, and then take the sum of all the CPU resource of each procedure for VNF_j . The memory resource

Memory/CPU(GB)	1 CPU	2 CPUs	3 CPUs	4 CPUs	5 CPUs	6 CPUs
Function1	15.1/16.6	8.6/8.7	12.9/13.1	12.3/12.5	10.4/12.2	10.3/10.8
Function2	23.6/24.0	16.6/19.9	13.4/13.6			
Function3	5.4/6.2	7.48/7.54	6.3/7.1	5.5/5.8		
Function4	2.7/3.1	3.7/3.8	3.2/3.5	2.7/2.9		
Function5	8.9/9.0	5.7/7.2				
Function6	4.4/5.0	2.54/2.63				

Table 1: Through three rounds at the Muni.

can be derived in the same way. As for network payloads, we can simulate the total size of network throughput by weighted sum of payload profiles.

4 Resource Allocation

Given information of the state transition probability matrix P_{ij} , the payload profile matrix L_{kl} and the state-VNF mapping matrixM_{ij}, we can make another intermediate transition matrix T_{ij} to manage the resources reallocation.

4.1 Resource Saving

Although we know the resource we need, but there are still problems to allocate the resource because of the different demands of the functions. For the test devices in our network, every CPU have 8G memory in average, which is usually not enough for some functions though maybe too much for the others. In the table, the number is minnum/maximun, which means in most situations, if the required cpu is known, the memory demands won't change in a large range. It seems that picking the functions in proper pairs can make the memory usage more efficient according to the relationship. But it fails in the transition process. Because of the big change of the average memory cost per cpu, we may have to choose process migration or waiting for the upper layer to give their task schedules.

4.2 Payload balancing

When changing the resources we need, We hope to allocate the resources as a queue instead of a stack. As there always be some resources rested, we hope

that our server can take a break in a round turn. Then comes an algorithm, we treat the working devices as a queue. When we need more resources we can include the server on the head of the queue while return the rest resources from the tail of the queue.
However, we can't do that. As a service is runing, we can't stop it and make it run at another device immediately(it's about process migration, which can be easily found on commercial Linux release). However, we can only migrate the VM alive within 6s. Though it performs very well, it's still long enough to make some influence on our service. It is OK for the functions which consume much more resources for the great changes in their usage can easily be used to make the VMs work in turns. But for some function, they always need one cpu and some memory changes in a little range. We can't let the VM relax without migration unless we can use another VM run the tasks comes lately and wait for the previous tasks finished.

4.3 Progress Migration

There are there common algorithm about the process migration, which is similar to the VM migration – the core of the algorithm is to copy the data from one device to another. Obviously, migrating the process is much more sensible except that to realize progress migration, some changes about the kernel must be done.
Luckily, I find a program 'Linuxpmi'which can make a linux-2.6 kernel(It has been long time before it is last updated). With this kernel, we can create new kvm VMs. However, to completely solve the problem, more work must be done about how to deal with the kernel and add process migration into it.

4.4 Controlling the Group

There is a gap between that we know hoe to allocate the resources and that we know how to change the current group state into the target one. It is a boring and tiring job and I won't introduce it in detail.
We have two code to control the VMs, one in python and the other in Linux shell. They records the detailed commands to operate the VMs and we can use it with some logical statements as "open slave2 vm3".
With all the work above, we can truly make the usage of the sources very close

to the ideal allocation method (with a little dalay).

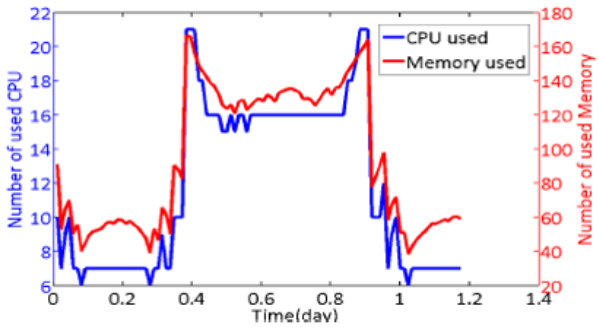


Figure 4: Ideal resources usage after allocation

4.5 Supposed solution for resource allocation

In reality, the current algorithm we realized can't live up to our expectation for we haven't realize the process migration. It would be better if we can migrate the process with very low cost.
However, there is an approximate simple method. Though, we can't make all the VMs takes the same burden but we can let them exchange their work from time to time. For example, we can migrate the VMs when we do some regular maintenance. In a large range of time, their Payload can be quite similar.

5 Summary and the

The main work of testbed is allocate the resource. After get the number of user and the user behavior model, we can calculate the resource demand. Then we adjust the current resource state into the needed state. At first, this part of work is just a manual process. We modified the cluster and realize the resource allocation regardless the reality strict. Process migration is necessary for the general solution of our allocation method. At the same time, the task scheduling is also ignored as we don't have the program to run the process. Once gets it, specific solution for payload balance may be developed based on the task side.