An Open NFV and Cloud Architectural Framework for Managing Application Virality Behaviour

Dilip Krishnaswamy
IBM Research
Bangalore, India
dilikris@in.ibm.com

Ram (Ramki) Krishnan
Brocade Communications Systems, Inc.
San Jose, CA
ramk@brocade.com

Diego Lopez
Telefonica I+D
Madrid, Spain
diego.r.lopez@telefonica.com

Peter Willis
British Telecommunications
Adastral Park, England
peter.j.willis@bt.com

Asif Qamar
Evolv
San Francisco, CA
asif@asifqamar.com

Abstract—One of the key goals of Network Functions Virtualization (NFV) is achieving energy efficiency through workload consolidation. A good example for maximizing energy savings is the Virtualization of Content Delivery Networks (vCDNs) NFV use case where the video streaming workloads exhibit significant difference between prime-time and non-prime-time usage of the infrastructure. This paper examines the practical challenges in maximizing energy efficiency for vCDN workloads. This paper proposes an open NFV architectural framework for conveying content virality information from Cloud applications such as YouTube, Twitter and mechanisms for leveraging it to maximize the energy efficiency for vCDN workloads. This paper also proposes a more general architecture for any Cloud/NFV application that may experience virality.

Keywords— Data Center, CDN, NFV, Energy Efficiency, Workload Consolidation, Application Virality, Viral Video, Virtual Machines, Resource Provisioning, Resource Management

I. INTRODUCTION

Network Operators use a variety of proprietary hardware appliances. Hardware appliances deliver good performance typically. However, they are complex to manage, and not easy to scale up/down in capacity and not cost effective. NFV [1] is a movement by network operators around the world to address the aforementioned issues. NFV involves the implementation of network functions in software that can run on a range of industry standard server hardware, and that can be moved to, or instantiated in, various locations in the network as required, without the need for installation of new equipment. NFV has many use cases [2], notable of which is the vCDN. The goal of vCDN would be to address virtualization of all the CDN components, but the biggest and immediate impact would be on the cache nodes given the growth in content especially in mobile networks [3]. Figure 1 depicts three instances of vCDN cache nodes each running on multiple Virtual Machines (VMs).

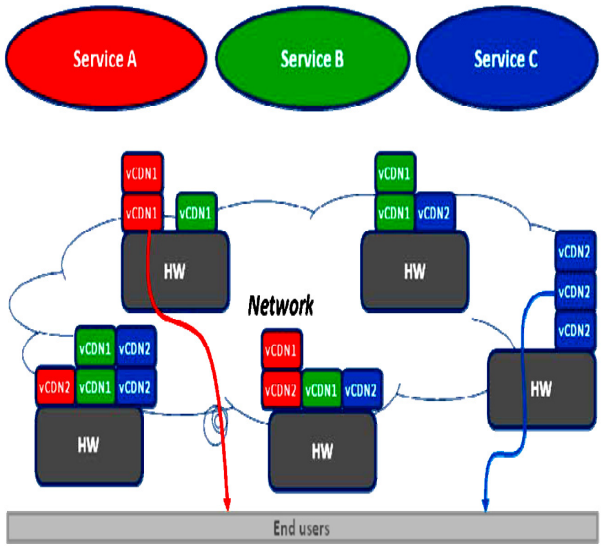


Figure 1: Virtualization of CDNs
(Source: ETSI NFV Use Cases [2])

The key benefits of vCDNs are

- Overall capacity is shared by all content delivery appliances and other virtualized appliances.
- Since appliances are pure software, it is easy to replace or modify them in the event of new CDN requirements.
- Besides caching of operator's own content, this enables operators to offer content caching as a service to CDN providers (e.g. Akamai Aura CDN) and large content providers with private CDNs (e.g. Netflix OpenConnect).

Currently, the allocation of VMs for vCDN follows a static model based on weekday prime-time characteristics, business hours etc. This model results in substantial resource over-provisioning, since a lot of content viewed over websites like YouTube and shared over social media like Twitter follow a virality pattern during anytime of weekday or weekend [4], as

978-1-4799-6390-4/15/$31.00 ©2015 IEEE

depicted in Figure 2. Additionally, many industry standard servers consume substantial power in the active idle state, which results in severe energy inefficiency. For example, HP ProLiant DL380p Rack Server has a peak power utilization of 324 Watts and consumes 105 Watts (approximately 33% of peak) in the active idle state – an exemplary depiction is in Page 17 of [5].

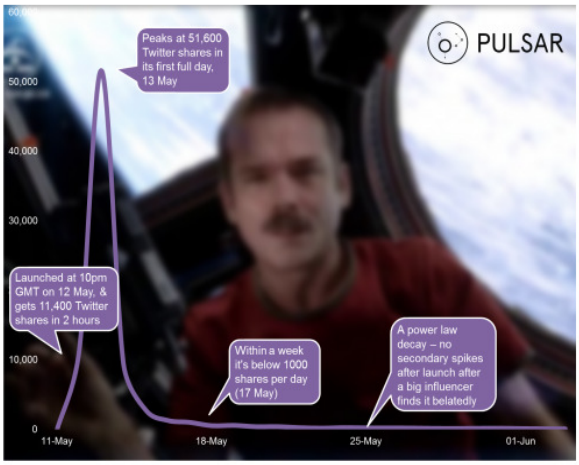


Figure 2: How Videos go Viral
(Source: Facegroup.com [4])

One of the key goals of Network Functions Virtualization (NFV) is achieving energy efficiency through workload consolidation [6]. This paper proposes an open NFV architectural framework for conveying content virality information from applications such as Twitter, Facebook, YouTube and mechanisms for leveraging it to maximize the energy efficiency for vCDN workloads without compromising performance. This enables network operators to offer new types of content caching services to its CDN customers related to managing viral content and to also optimize the resource usage of their own CDNs. This paper also includes a performance comparison of the proposed approach with a static model of allocation of VMs for vCDNs and demonstrates the benefits of a more dynamic approach. Besides vCDNs, this paper also proposes a more general architecture for any network applications that may experience virality.

Related research work focused on video caching is described in [7] and [8]. [7] addresses energy efficiency in CDNs using dynamic power management techniques; our approach does not rely on dynamic power management capabilities in the server platform. [8] describes a complementary approach to energy efficiency of video caching systems through appropriate selection of either single-level or multi-level caching based on different end-user behavior similarities.

II. Open NFV Architecutural Framework

Figure 3 depicts the Open NFV Architectural Framework, adapted from the definition of the ETSI NFV Architectural Framework [9] and extended in this paper to show support for content virality management. It has the following virtual and physical or hardware (HW) infrastructure components as part of NFV Infrastructure (NFVI)

1) Compute - physical servers hosting computing elements in the form of Virtual Machines (VMs)
2) Storage – physical/virtual storage
3) Networking – physical/virtual routers.

The Virtual Infrastructure Manager (VIM), which could be accomplished by open source software like OpenStack [10] for example, performs lifecycle management of the above infrastructure components and maintains a dynamic resource pool of the same. Virtual Network Functions (VNFs) such as firewall, load balancer, CDN etc., each of which runs on multiple VMs, are managed by Virtualized Network Function Manager (VNFM), which performs lifecycle management of VNFs and maintains dynamic resource pool(s) for different types of VNFs. The VNFM exchanges Virtual/Physical resource information with the VIM.

The other elements of the NFV architectural framework include a service orchestrator, and management and support systems such as an Element Management System (EMS), an Operations Support System (OSS), and a Business Support System (BSS).

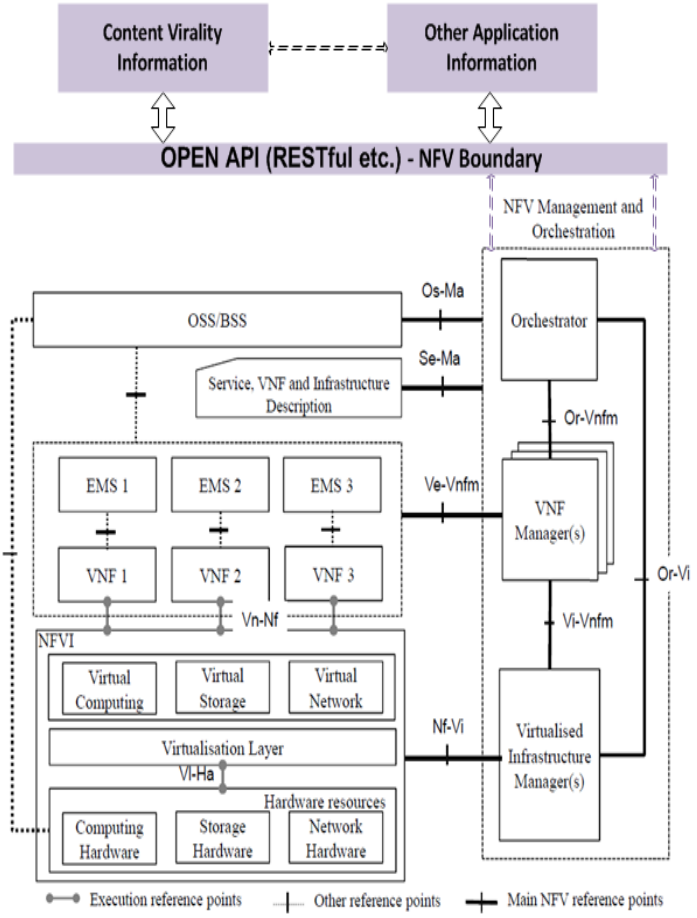


Figure 3: Open NFV Architectural Framework
(Adapted from [9])

978-1-4799-6390-4/15/$31.00 ©2015 IEEE

The various interfaces in the Open NFV architectural framework are

- Vi-Ha – Interface between the virtualization layer (e.g. hypervisor for hardware compute servers) and hardware resources
- Vn-Nf – Represents the execution environment provided by the NFVI to VNF (e.g. a single VNF could have multiple VMs)
- Nf-Vi – Interface to the VIM – used for VM lifecycle management and other purposes
- Ve-Vnfm – Interface between VNF/EMS and VNF Manager – used for VNF lifecycle management and other purposes
- Se-Ma - Used for getting information about VNF deployment template and other purposes
- Os-Ma – Interface to OSS/BSS - handles network service lifecycle management and other functions.
- Vi-Vnfm – Interface between VIM and VNFM – handles resource allocation requests by the VNF manager and other functions
- Or-Vnfm – Interface between Orchestrator and VNFM – handles collection of state information necessary for network service lifecycle management and other functions

To support content virality information in this open architecture, we suggest the availability of such information through open interfaces as depicted in Figure 3. OpenAPIs could be designed to exchange information across open interfaces, which could be based on a RESTful framework [11] or a pub/sub framework. Content virality information can be streamed in real-time from applications such as YouTube, and submitted to the VNFM through open APIs. This information can be used by VNFM to populate the newly allocated vCDN resource pool with the optimal VNF capacity needed for content caching which can be consolidated to a minimal set of VMs and physical servers. Besides content virality information, we also suggest that the architecture could optionally provide a generic open API framework for handling other application information, such as information regarding firewall services, in real-time if available.

In typical current systems, the vCDN resource pool is statically populated by policies such as weekday prime-time characteristics, business hours etc., and could be significantly over-provisioned to handle any dynamic requests. Current systems also do not delve into specific targeted use cases or a framework for conveying application information in real-time.

The rest of the contribution of this paper is to develop these aspects further in an open architecture framework as suggested in Figure 3. In effect, the differentiating aspects of the proposed architectural framework in this paper are

- A dynamic resource pool that is used to optimally populate the vCDN VNFs with the right amount of VMs and physical servers to minimize over-provisioning.
- Parameters of interest for real-time streaming of application information such as content virality, that could be utilized for resource optimization in an open-API framework

III. System Analysis

The virality pattern of videos shared over social media like twitter typically follows a distribution that has a sharp rise in an early phase, followed by a slow decay phase. In some cases, the sharp rise is quickly followed by a sharp decay. The aggressive sharing across social media results in a multiplicative effect of information propagation that results in a fast rise in terms of accesses. For video access over YouTube [12], one can expect viral behavior to follow a pattern such as the one depicted in Figure 4 shown for the daily access statistics for the *PSY Gangnam Style* video [13] or the *Waka Waka (This Time for Africa)* video [14] in Figure 6.

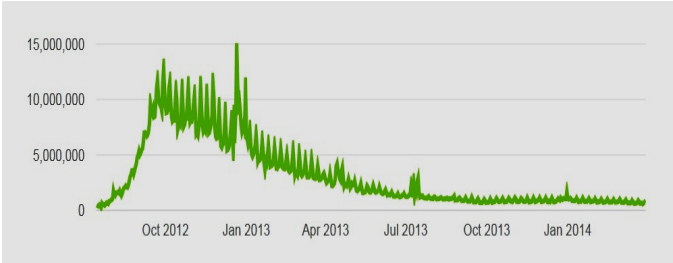


Figure 4: *PSY Gangnam Style* Video Daily Access Stats [13]

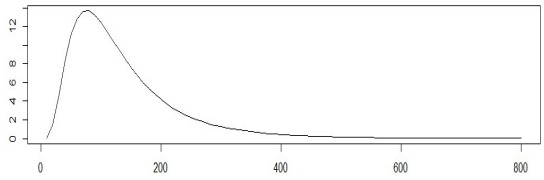


Figure 5: *PSY Gangnam Style* Video Access Primary Envelope
Y-Axis – Daily Accesses in millions

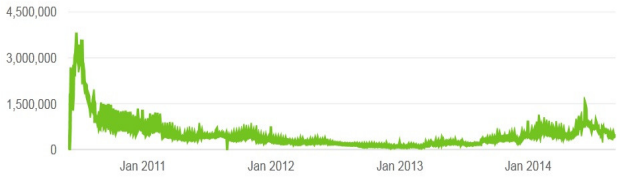


Figure 6: *Waka Waka (This Time for Africa)* Video Daily Access Stats [14]

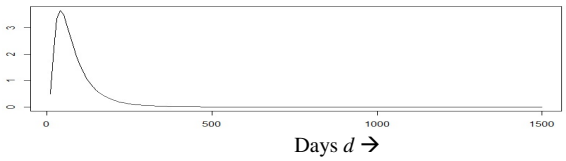


Figure 7: *Waka Waka (This Time for Africa)*
Video Access Primary Envelope
Y-Axis – Daily Accesses in millions

For such videos one can observe an overall primary envelope functional form for the video accesses as function of time,

978-1-4799-6390-4/15/$31.00 ©2015 IEEE

where the envelope functional form can be partitioned into two phases:
1) Phase I: An initial sharp viral rising phase, followed by
2) Phase II: A slow decay phase that persists for a long time as the interest in the video fades away over time.

Such functional behavior can be described using a family of generalized extreme-value distributions [15]. The approximate max-likelihood-estimate functional form for daily access for a specific video as a function of time, can typically take the form of a scaled log-normal or weibull or gamma distributions based on actual temporal profile of the virality. For example, the approximate primary envelope function in Figure 5 for the video access stats in Figure 4 can be described by a log-normal function [15] of the form

$$f_1(d) = \frac{2000}{0.63\, d\, \sqrt{2\pi}} e^{-\frac{(\ln d - 4.725)^2}{2*0.63^2}} \quad \text{.....(1a)}$$

Similarly, the approximate primary envelope function in Figure 7 for the video access stats in Figure 6 can be described by a log-normal function [15] of the form

$$f_2(d) = \frac{330}{0.7\, d\, \sqrt{2\pi}} e^{-\frac{(\ln d - 4.19)^2}{2*0.7^2}} \quad \text{.....(1b)}$$

One can also observe that there is typically some random noise around the envelope which can vary over time. Let us assume that such noise is normal of the form $n_0(d) = N(\mu_0(d), \sigma_0(d))$ with mean $\mu_0(d)$and variance $\sigma_0(d)$. Then an envelope function $R_0(d)$ for daily access statistics with a primary envelope function $f_0(d)$ and an additional noisy component $n_0(d)$ can be represented as

$$R_0(d) = f_0(d) + n_0(d) \quad \text{.....(2)}$$

In addition, one can observe that the number of daily accesses as a function of time can have additional secondary ripple fluctuations due to the nature of viral video propagation through social media. These additional ripples R_i can be scaled-down versions of the original function $R_0(d)$ or come from a family of extreme-value distributions that can be similar in form to the primary envelope, but delayed in time. Then the overall daily accesses can be represented as a summation of a number of scaled noisy envelope functions shifted in time. This could be written as a function $S(d)$, given by

$$S(d) = R_0(d) + \sum_i \beta_i R_i(d - \delta_i) \quad \text{.....(3)}$$

where $$R_i(d) = f_i(d) + n_i(d) \quad \text{.....(3a)}$$

For example, for Figure 4, this summation reflects the additional ripple effects observed around Jan 2013, and August 2013 that are superposed around a primary envelope function for the access statistics. The factor δ_i reflects the shift in time when the i^{th} viral ripple effect occurs, and the factor β_i reflects an attenuation factor on the ripple relative to the primary envelope. The form of the ripple function R_i can also be expected to vary amongst a family of distributions. The video access stats in Figure 6 is specifically interesting. Although the primary envelope had a sharp rise in 2010, and decayed by 2011, additional viral ripples increased accesses relative to the primary envelope, particularly in 2014, around the time that the next football world cup was played, but were of a diminished magnitude (factor β_i) relative to the primary surge in 2010. It can be observed that while there can be a general envelope pattern to the access statistics for a viral video, the actual number of accesses can vary dynamically around the envelope function. If resources need to be planned in a data center to deliver the content related to such a video, then an average number of resources could be planned based on an expected envelope function. However, it is desirable to plan for additional viral information for unpredictable ripple events R_i (which can themselves have noisy components) around the envelope R_0, or the degree of noise variability $n_0(d)$ around the primary envelope $f_0(d)$.

From a dynamic resource planning perspective in a cloud data center, it is desirable to understand the number of accesses in the near past, and then make a prediction of the expected number of accesses in the near future, with minimal overprovisioning around an expected predicted access need for a given video. Different videos may be in different phases of virality, as some videos may be in a decay phase whereas other videos may be in a slower plateau phase, and some other new videos may be experiencing a sharp rise in demand for access. The proposed framework will allow the prediction of the expected access requirements for each of the videos, with an aggregate prediction of access requirements across all videos, where such information is exposed via open APIs to allow for resource planning and a refinement of the resource planning on a regular basis. Such refinement could happen on a weekly or a daily basis or on the order of few hours typically. Local analytics engines can estimate the general form of the access statistics $S_{j,v}(d)$ for a viral video v. This can be used to estimate the average number of VM resources

$$W_{j,v}(d) = S_{j,v}(d) / \lambda_{j,v} \quad \text{.....(4)}$$

Here $\lambda_{j,v}$ reflects the average number of videos that can be played by a single VM, and $W_{j,v}(d)$ is the number of VM resources required in a data center j based on the daily access requirements $S_{j,v}(d)$ for a given video content v. A data center can thus choose to use its own local estimate of usage to predict resource requirements for a given video for future days. Global statistics could also be used to determine the usage requirements at a data center as well. Global information across a network of data centers can be gathered to infer global daily access statistics $S_{global,v}$ for the video v, given by,

$$S_{global,v}(d) = \sum_i S_{i,v}(d) \quad \text{.....(5)}$$

Such global information can then be provided to a specific data center. The fractional usage $\alpha_{j,v}(d)$ at a data center j relative to the global usage can be estimated by using the relationship,

$$\alpha_{j,v}(d) = \frac{S_{j,v}(d)}{S_{global,v}(d)} \quad \text{.....(6)}$$

978-1-4799-6390-4/15/$31.00 ©2015 IEEE

Based on global expected mean usage information $S_{global,v,\ mean}$, and a most recent estimate of the fractional expected usage $\alpha_{j,v,mean}$, the mean expected usage $S_{j,v,\ mean}$ for a specific video content v at a data center j, can be given by $\alpha_{j,v}\ S_{global,v,\ mean}$. Based on such a mean estimate of usage $S_{j,v,\ mean}$, the number of resources required to support the content at the data center can be estimated based on the proportionality factor $\lambda_{j,v.}$

In addition to local and global estimates of content access information, one can also utilize information obtained through inter-domain or inter-data-center information exchanges through open interfaces to exchange such information. Thus a combination of local, global, and inter-domain aggregate measures can be utilized to predict resource requirements as depicted in Figure 9. Further discussion on utilization local, global, and inter-domain measures is provided in section V. The parameters shown in the figure are discussed below. Based on the number of contributing domains, an aggregate statistics across contributing domains can be obtained such as the accumulation of information for global statistics in equation 5. However, such aggregation can be performed locally. Thus a data center j can infer aggregate daily access statistics $S_{aggregate,v}$ for the video v, over data centers or domains k providing information given by,

$$S_{aggregate,v}(d) = \sum_{k} S_{k,v}(d) \quad \text{.....(7)}$$

The fractional usage $\eta_{j,v}(d)$ at the data center j with respect to inter-domain / inter-data-center aggregate statistics relative to the aggregate usage can be estimated by using the relationship,

$$\eta_{j,v}(d) = \frac{S_{j,v}(d)}{S_{aggregate\ ,v}(d)} \quad \text{.....(8)}$$

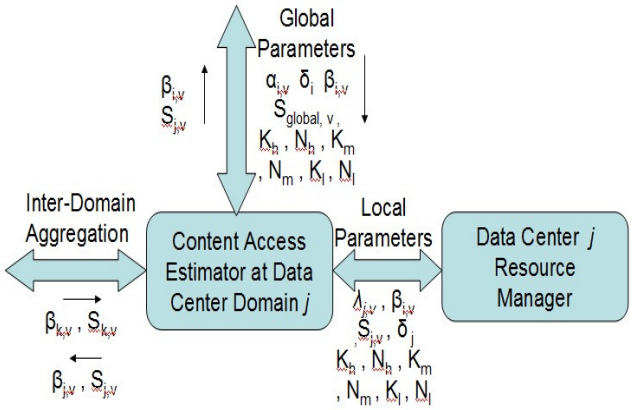


Figure 8: Content Access & VM Resource Estimation

The choice of using location estimation and/or leveraging global statistics and/or inter-domain aggregation information to estimate local usage can be left as an implementation choice as shown in Figure 8. In general a weighted combination of the local/inter-domain/global measures can be used, and the content access estimator can refine its estimate $S_{j,v,est}$ by appropriately weighing its estimates.

$$S_{j,v,est} = \theta\, S_{j,v} + \rho\alpha_{j,v} S_{global,v} + \varphi\eta_{j,v} S_{aggegate,v} \quad \text{.....(9)}$$

where θ, ρ, and φ are weights chosen such that $\theta + \rho + \varphi = 1$. Information related to the number video accesses can be stored in a local database or memory, and retrieved as needed to estimate data center resource requirements. Such information could be provided using open interfaces to exchange information between a content access estimation engine and a resource management engine on a data center platform. In general, a data center j will need to worry about multiple videos. Let us now assume that such expected usage information for various viral videos is available at a given data center. For each video v, let the expected number of accesses be $S_{j,v}$. Then, the total number of VM resources W_j required for all the videos would be given by

$$W_j = \sum_{v} \frac{S_{j,v}}{\lambda_{j,v}} \quad \text{.....(10)}$$

Since the number of viral videos could be potentially very large, it might be better to classify videos into categories based on usage, and address resource requirements for each class, and subsequently aggregate resource requirements across all classes. In the extreme case, each video could be considered to be in a class of its own. Let us assume that the total number of accesses for a particular class of videos, across all videos in that class) be $K_{j,i}$. Then

$$K_{j,i} = \sum_{v(i)} S_{j,v(i)} \quad \text{.....(11)}$$

where summation over $v(i)$ represents all videos in class i.
Let us assume that there are $N_{j,i}$ videos in class i. If $\lambda_{j,i}$ represents the average number of videos of class i that can be played by a VM, then the average number of VMs $W_{j,i}$ required in a particular day to support the accesses of class i is given by

$$W_{j,i} = \frac{K_{j,i}}{\lambda_{j,i}} \quad \text{.....(12)}$$

Then the total resources W_j required by data center j across all classes i is given by

$$W_j = \sum_{i} W_{j,i} \quad \text{.....(13)}$$

If the above measures are based on daily access statistics, then the expected resources $w_j(\Delta t)$ for a given interval of time Δt hours relative to the overall duration of a day, would then the computed as a fraction of the resources estimated for the entire day given by $w_j(\Delta t) = (\Delta t/24) * W_j$. In general, if the number of expected VM resources can be predicted well, then servers that are not required can be turned off resulting in improved energy efficiency in the system as well.

Let us take an example with 3 video classes. In this case, let us classify videos into multiple classes such as H (High), M (Medium), or L (Low). The number of accesses S_i for a video v_i that crosses a significant threshold greater than T_{high} accesses per day would fall into the class H. The number of accesses S_i for a video v_i in the class M would cross a medium threshold of T_{medium}, but would be lower than T_{high}. The class L of videos would constitute videos that where the number of accesses would be lower than T_{medium} but higher than a lower

978-1-4799-6390-4/15/$31.00 ©2015 IEEE

limit of $T_{low.}$ (where $T_{low.}$ could be set to 0 or some non-negative lower threshold).

Class H : $S_i > T_{high}$

Class M : $T_{medium} < S_i \leq T_{high}$

Class L : $T_{low} < S_i \leq T_{medium}$

Videos in Class H would typically have the most significant impact to the resource provisioning requirements in the data center. Correspondingly, videos in class M or L would typically have progressively less significant impact on the resource provisioning requirements in the data center. However, the number of such videos may be very small in class H such as in the order of a few tens, whereas videos in the class M can be higher such as in the order of a few hundreds. The rest of the reasonably significant videos that exceed the lower threshold T_{low} would fall into class L and can be significantly larger in number with lower impact on a per video basis. Let the dynamic expected total number of accesses per hour for classes H, M, and L be K_h, K_m, and K_l respectively, and let N_h, N_m, and N_l be the number of videos in classes H, M, and L respectively. Then the expected number of accesses $S_{total,dynamic}$ that need to be provisioned per hour for would be given by

$$S_{total,dynamic} = K_h + K_m + K_l \quad \text{.....(14)}$$

Based on recent accesses for the videos, an aggregate requirement for the entire set of videos can be planned for. For example, let us assume that the system is operating in a mode (at a time of the day when the viral video consumption may be significantly lower than a peak level of consumption) such that the dynamic requirements for viral video support is given as shown in Table 1. by K_h = 8.000,000, N_h = 16, K_m = 6,000,000 , N_m = 120, and K_l = 5,000,000 and N_l = 1000. Then,

$$S_{total,dynamic} = K_h + K_m + K_l = 19{,}000{,}000$$

Let us assume that the number of videos that can be served per VM per hour be λ_{ave} . λ_{ave} can be an average estimate over $\lambda_{j,v}$ values across videos v at a data center j. In this example, λ_{ave} = 200,000 (see vCDN test setup below for more details). Then the number of vCDN VMs required is given by

$$W_{total,dynamic} = S_{total,dynamic} / \lambda_{ave} \quad \text{.....(15)}$$

In this example, $W_{total,dynamic}$ =19,000,000/200,000 = 95.

Now, the videos in class H can have a high degree of variance in their accesses, whereas videos in class M can have lower variance, with videos in class L having the least variance. For example, a static estimate of accesses could provide higher and more conservative estimates such that, for example, $K_{h,\ static}$ = 25,000,000, $K_{m,static}$ = 9,000,000 and $K_{l,static}$ = 9,000,000, in terms of expected total number of accesses per hour usage. Then the max total number of accesses that the system would need to plan for without dynamic information about usage requirements could be given by

$$S_{total,static} = K_{h,static} + K_{m,static} + K_{l,static} \quad \text{.....(16)}$$

Then the number of vCDN VMs required is given by

$$W_{total,static} = S_{total,static} / \lambda_{ave} \quad \text{.....(17)}$$

In the aforementioned example,

$S_{total,staticx}$ = 43,000,000

$W_{total,static} = S_{total,static} / \lambda_{ave}$ = 215

In a statically provisioned system, one would have to typically plan for such a worst case for the maximum expected number of resources, which would be given by 43 million accesses or equivalently 215 VMs. However, if the system can be dynamically provisioned based on expected needs on a daily basis or a k-hourly or hourly basis for example, then one could plan for the number of resources to be much closer to the expected number of accesses for the day. In addition, the number of accesses can be dynamically overprovisioned by a small factor γ which would increase the number of accesses that need to be dynamically provisioned to $S_{total,dynamic} * (1 + \gamma)$. Thus, the number of vCDN VMs needed including an overprovisioning factor of γ is $W_{total,dynamic} * (1 + \gamma)$. For the given example with γ = 0.1, this would be 105. Therefore the overall reduction in the number of VMs that need to be provisioned is given by

$$W_{diff} = W_{total,static} - W_{total,dynamic} * (1 + \gamma) \quad \text{.....(18)}$$

For the given example with $\gamma = 0.1$, this would be 110 vCDN VMs that are not activated at all resulting in corresponding energy savings. These calculations for the above example are depicted in Table 1 and Table 2. Thus, with dynamic information propagation in the system, a significant number of servers can be turned off with reduced provisioning in the system, which can help with significant energy savings in the system. Alternatively, the unused servers can be used for running other workloads especially the delay-tolerant category such as Hadoop. The aforementioned framework could be extended to include other types of content besides video as well.

Operational Mode	K_h	K_m	K_l
Typical (static)	25,000,000	9,000,000	9,000,000
Proposed (dynamic)	8,000,000	6,000000	5,000,000

Table 1: Accesses for video classes - static vs dynamic

Operational Mode	Total Number of Accesses per hour	Number of video accesses per vCDN per hour	Total number of vCDN VMs
Typical (static)	43000000	200000	215
Proposed (dynamic)	19000000	200000	105

Table 2: Number of vCDN VMs – static vs dynamic

The Physical server/VM configuration for the vCDN tests conducted by British Telecom (BT) is as follows.

Physical Server HP BL460c G8 configuration

- 2 sockets, 8 cores per socket (Intel Xeon E5-2680 @ 2.7GHz), 130G RAM, 146G Spinning Disk, 800G SSD
- 2 * 10GigE NIC Intel 82599EB (2-port 560FLB Adapter)
- 16 vCPU partitions (one per core)

vCDN VM configuration (one per physical server)

978-1-4799-6390-4/15/$31.00 ©2015 IEEE

- 12 vCPU partitions, 40G RAM, 140G HD (for OS on Storage Array), 720G HD (SSD cache storage, allocated in 4 blocks of 180G each)

Approximate power utilization per vCDN VM (Note 1)
- Peak (100%) utilization – 300W
- Active Idle (approx. 33% of peak utilization) – 100W

It should be noted that the tool HP power calculator [16] was used to calculate the power utilization per blade chassis for the above configuration; this is used to calculate the approximate power utilization per vCDN VM.

For an Adaptive Bit Rate (ABR) ABR Video on Demand (VoD) test scenario using say MPEG-DASH [17], due to differing network constraints across users, a distribution of users with differing bandwidths will result in a distribution of load created by the users on the physical system. The load distribution across users results in an aggregate load on the system. We will assume that on an average, a total of V video accesses per hour can be supported by the above vCDN VM configuration based on the aggregate load and the load distribution across users, average video viewing durations, and available core/memory performance in the system. For the example considered, the aggregate bandwidth of each vCDN VM was 10Gbps supporting 10,000 concurrent streams/users on average. For an average viewing duration of 3 minutes (=1/20 of an hour) for each video access, the system can support λ = 200,000 video accesses per hour on average.

Operational Mode	Physical servers at 100% capacity consuming peak power	Physical Servers in active idle state	Peak Power usage per physical Server (W)	Idle power usage per physical server (W)	Total Power Consumption (W)
Current (static)	95	120	300	100	40500
Proposed (dynamic)	95	10	300	100	29500

Table 3: Energy consumption – static vs dynamic

In a statically provisioned model, a total of $W_{idle,static}$ servers will be in active idle state consuming energy. Let us assume that these servers are consuming power P_{idle}. Then these servers will consume an aggregate power $P_{idle} * W_{idle,static}$. For the given example, 120 servers will be in active idle state consuming approximately 33% of peak power – an exemplary depiction is in Page 17 of [5]. In the current dynamic proposed model, only $S_{idle,dynamic}$ = 10 servers (these are the over-provisioned servers) will be in active idle state typically, which will consume an aggregate power of $P_{idle} * W_{idle,dynamic}$. The rest of the remaining unused servers W_{diff}, where

$$W_{diff} = W_{total,static} - W_{total,dynamic} * (1 + \gamma)$$
$$= W_{idle,static} - W_{idle,dynamic} = (120 - 10) = 110$$

will be powered off and will not consume any energy. Thus, for the given example, the average power savings with the proposed dynamically provisioned model for this illustrative example is given by

$$W_{diff} * P_{idle} = (120 - 10) * 100 = 11000 \text{ Watts.}$$

These calculations are depicted in Table 3. From Table 3, the percentage energy savings of the dynamic (proposed) vs the typical static (current) scheme is given by (11000/40500)*100 ≈ 27.16%. An available approach is to power up/down physical servers on demand based on need for additional capacity. Typically, it takes a few minutes to boot up servers and bring up a virtualized environment, impacting server availability. It is therefore good to overprovision servers marginally and keep them operating in an idle mode, to avoid impacting user experience on account of server unavailability when needed.

IV. Open API Parameters

A. Content Virality Information

It would be desirable to create open interfaces to exchange parameter information such as hourly class requirements K_h , N_h , K_m , N_m , K_l , N_l for the different classes of video, average resource requirement λ_{ave} , over provisioning factor δ_j for data center j, and/or specific statistics for high profile videos $S_{j,v}$ and $\lambda_{j,v}$. When using specific video statistics for high profile videos, the class information for different video classes should exclude requirements for these specific high profile videos, so that the aggregate information across classes and the specific videos can be utilized to determine overall requirements. An open API such as one based on a RESTful framework [11] or a pub/sub framework can be designed that reflects the requirements for viral video processing. All such factors can be exchanged through open interfaces and can help in the dynamic determination of resources that would need to be provisioned. When global statistics are used over global open interfaces, a fractional correction factor $\alpha_{j,v}$ would be needed as well, to reflect the fraction requirements for a specific data center. Additionally, the following information could be useful for further optimization

1) Additional Video/content meta-data such as sizes or formats of highly relevant viral videos/content to provision for storage resources

2) The expected usage for video at different video data rates, e.g. SD/HD, within each video class. This can be provided for different levels of video as proposed in the High Efficiency Video Coding (HEVC) standard [18].

3) The expected usage of other types of content, besides video, that can have a significant impact in terms of energy consumption, so that resource planning can be enabled for such information.

4) The expected usage of individual videos including relevant content metadata information including new viral videos, or viral ripple events of existing content especially related to videos of class H that have a large number of daily accesses per video.

The above information could be conveyed from any public or private source of video virality to the network operator who is offering virtualized CDNs as a service to CDN providers and large content providers.

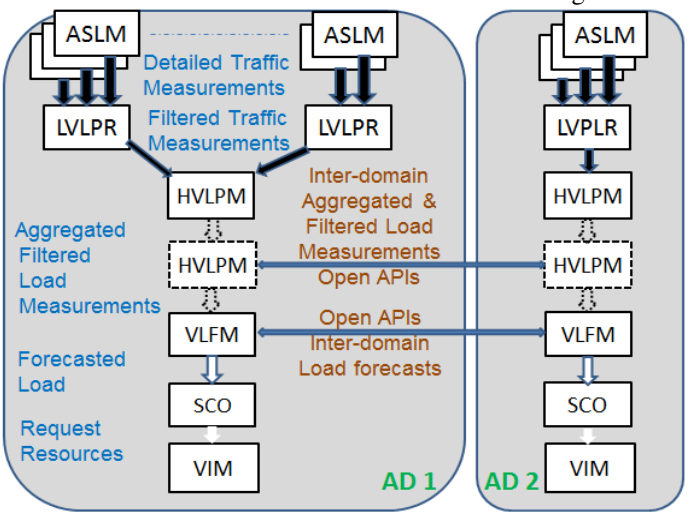
978-1-4799-6390-4/15/$31.00 ©2015 IEEE

B. *Other Application Information*

Besides content virality information, the open API can be used for conveying other application information. For example, the NFV network operator could be offering firewall as a service to its enterprise customers. By conveying information about the categories of enterprise applications and usage during different times of the day, the NFV network operator has an opportunity to optimize the resources used by the firewall and deliver valued added firewall services to its enterprise customers.

V. Open Virality Architecture for Generic Cloud and NFV Applications With Inter Domain Aggregation

In general, other applications may also exhibit virality behavior for example; incoming voice calls to a call center, video of live events, purchasing transactions, financial trading, online gaming, real-time maps. Just like CDN, applications like online gaming and real-time maps benefit from running in the NFV DCs in the edge of the network. It is therefore useful to make the open architectural framework more generic. Generally, a network service that may be impacted by virality behavior will be deployed and managed as a service chain for example VoIP calls to a call center will require a chain of various virtual network functions such as routers, Session Border Controllers, SIP servers and associated load balancers. Another example is the introduction of elements guaranteeing online game fairness, so all players have the same experience in despite the particular conditions of their network: An unfair game quickly becomes extremely boring even for winners. With static provisioning each component in the service chain will be provisioned to handle maximum load, what is completely impossible to predict when it comes to viral behavior. With dynamic provisioning, the resources are adjusted to match demand, however there are limits to how quickly the resources can be turned up and down, and a viral event may not be handled optimally in a normally heavily damped provisioning system e.g. the system may adjust resources every day based on long term growth over the year. With the incorporation of a virality model in the provisioning system it should be possible to predict the steep increase in resources required in the short term without over provisioning in the long term. However forecasts based on localized measurements may be less reliable and the normal statistical noise may accidently trigger provisioning for a viral event. In this regard, it would be good for these forecasts to be based on many distributed measurements where possible. A centralized entity could collate results from distributed entities, and then redistribute the aggregate global statistics. However, we could reduce dependence on the centralized entity by aggregating information across nearby domains. In that regard, one could explore inter-domain exchange of information through openAPIs so that inter-domain aggregation can be performed to better estimate resource requirements. This can help in refining a purely local estimate based on local measurements without needing global information. In general as suggested earlier in equation 9, a weighted combination of local, global, and inter-domain-aggregate measures can be used to determine resource needs. It is possible that the service chain crosses multiple administrative domains therefore the measurements are made in different administration domains from where the service chain resources need to be provisioned. Some administrations may not wish to expose the details of the measurements they have made and generally information about an individual's traffic cannot be disclosed. An architecture is required that allows hierarchical summary of virality information to reduce measurement traffic and hide detailed information. This architecture is shown in Figure 9.



Legend:
AD – Administrative Domain
VIM – Virtual Infrastructure Manager
SCO – Service Chain Orchestration
VLFM – Viral Load Forecasting Model
HVLPM – Hierarchical Viral Load Pattern Recognition
LVLPR – Local Viral Load Pattern Recognition
ASLM – Application specific Load Monitoring

Figure 9: Open Virality Architecture for Generic Network & Cloud Apps

Viral load pattern recognition software is run at the location where the applications loads are monitored, enabled by NFV. At this level pattern recognition may be a very broad filter and on its own could generate false positives. A series of hierarchical viral load pattern recognition functions are used to aggregate information from distributed sources and to distribute it to Viral Load Forecasting Models. These convert the current filtered load information into a short term load forecast. Load forecasts may be shared between administrative domains to improve forecast integrity. One important consequence of the application of NFV is that these pattern recognition functions can be virtualized themselves, and instantiated on demand to collect further evidence. Starting with a basic deployment of (not many) pattern recognizers, whenever a viral behavior is presumed, specific functions can be dynamically deployed in order to make a more informed decision. The Service Chain Orchestration function, part of the NFV Orchestrator functionality shown in Figure 3, will convert the forecasted load into resource allocation for each VNF. This information can be used by VNFM to populate the newly allocated resource pool with the optimal VNF capacity

978-1-4799-6390-4/15/$31.00 ©2015 IEEE

needed for the application/event which can be consolidated to a minimal set of VMs and physical servers.

VI. SUMMARY

This paper proposes an NFV architectural framework for conveying content virality information from Cloud applications such as YouTube, Twitter, Facebook and mechanisms for leveraging it to maximize the energy efficiency for vCDN workloads without compromising performance. A content access and VM resource estimator module is suggested as shown in Figure 8 that dynamically monitors intra-data-center usage along with inputs from global and inter-domain estimators, to dynamically predict an optimal resource requirement with a small degree of overprovisioning. Such a module could be an integral part of the data center or external to it as well, providing resource estimates to the data center. The paper also proposes a more general architecture for any Cloud/NFV application that may experience virality. Parameters have been suggested that reflect content virality information that could be exposed through open interfaces to enable resource planning to support such content. It was also shown that the availability and utilization of such dynamic information can significantly reduce the amount of VM resources needed. Such a framework can enable network operators to offer new types of content caching services their customers and to also optimize the resource usage, especially energy consumption, of their own CDNs. Optionally, instead of focusing on energy efficiency, the unused resources can be used for running other alternate workloads, especially in the delay-tolerant category such as background Hadoop applications.

VII. FUTURE WORK

The proposed framework could be extended further to learn content utilization characteristics of users for applications; this could include anonymized mobility patterns or viewing patterns of users or groups of users. Such information could be used to enable Cloud applications to improve their Quality of Experience (QoE) associated with content that is delivered to the end users. Machine learning techniques can be explored further to implement improved real-time intelligent control techniques for VNFM resource pool management. Additionally, information exchanged over social networks such as Facebook and Twitter about viral videos can be conveyed to NFV/cloud systems, which can be potentially used for pre-positioning of viral content in vCDNs for such systems. The authors hope that the proposed architectural framework suggested in this paper would also be relevant for use-cases such as JnanaEdge [19] to bring knowledge to the edge to serve remote / rural / under-served communities by learning the content that is relevant to such users, and delivering relevant viral knowledge and educational content to such users and communities using vCDN VMs in networks. In general, smart virality tracking and predictive techniques to optimally provision and utilize resources for data centers and improve energy efficiency of such data centers. Emerging open platforms for NFV could enable open interfaces for exchanging dynamic information to enable such optimization.

ACKNOWLEDGEMENTS

The authors would like to thank Gowrishankar Mandagradala (AMD) for discussions.

REFERENCES

[1] "ETSI NFV White Paper," http://portal.etsi.org/NFV/NFV_White_Paper.pdf http://portal.etsi.org/NFV/NFV_White_Paper2.pdf

[2] "ETSI NFV Use Cases," http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf

[3] "Cisco VNI White Paper: Global Mobile Data Traffic Forecast Update," http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html, February 2014

[4] "Facegroup: How Videos go Viral," http://www.facegroup.com/how-videos-go-viral.html

[5] "SPEC Benchmark Results: HP Proliant DL380p Rack Server," http://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Comparing-Dell-R720-and-HP-Proliant-DL380p-Gen8-Servers.pdf

[6] "ETSI NFV Virtualization Requirements," http://www.etsi.org/deliver/etsi_gs/NFV/001_099/004/01.01.01_60/gs_NFV004v010101p.pdf

[7] Bostoen, T. et al., "Minimizing Energy Dissipation in Content Distribution Networks Using Dynamic Power Management," Cloud and Green Computing (CGC), 2013

[8] Chien Aun Chan et al, "Energy savings dependency of IPTV caching systems on similarity in user behavior," 27th European Conference & Exhibition on Optical Communications, 2011.

[9] "ETSI NFV Architectural Framework," http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

[10] "OpenStack Open Source Software," https://www.openstack.org/

[11] Fielding, Roy Thomas, "Architectural Styles and the Design of Network-based Software Architectures," Dissertation, University of California, Irvine, 2000

[12] "Most viewed YouTube videos," http://en.wikipedia.org/wiki/List_of_most_viewed_YouTube_videos

[13] "PSY Gangnam Style Video,", https://www.youtube.com/watch?v=9bZkp7q19f0

[14] "Waka Waka (This Time for Africa)", The Official 2010 World Cup™ song, https://www.youtube.com/watch?v=pRpeEdMmmQ0

[15] Coles, Stuart, "An Introduction to Statistical Modeling of Extreme Values," Springer-Verlag, ISBN 1-85233-459-2, 2001

[16] "HP Power Calculator," http://h71019.www7.hp.com/ActiveAnswers/cache/347628-0-0-225-121.html

[17] "MPEG DASH Standard," http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57623

[18] G.J. Sullivan; J.-R. Ohm; W.-J. Han; T. Wiegand, " Overview of the High Efficiency Video Coding (HEVC) Standard," IEEE Transactions on Circuits and Systems for Video Technology, December 2012.

[19] D. Krishnaswamy, K. Rajagopal, A. Qamar, R. Krishnan, N. Narayan, "Bringing Knowledge to the Edge with JnanaEdge and BuddhiEdge: Architecture, Implmentation, and Future Directions," IEEE Technology For Education Conference, 2014, to appear.

[20] OPNFV, http://www.opnfv.org.

978-1-4799-6390-4/15/$31.00 ©2015 IEEE