Optimizing Cloud Resources for Delivering IPTV Services through Virtualization

Rezwana Begum¹, Dr. Mohd Abdul Waheed², Dr. Jayashree Agarkhed³

¹P.G.Student, Department of Computer Science & Engineering,
VTU RO PG Centre, Gulbarga, Karnataka, India.
²Associate Professor, Department of Computer Science & Engineering,
VTU RO PG Centre, Gulbarga, Karnataka, India.
³Professor, Department of Computer Science & Engineering,
PDA College of Engineering Gulbarga, Karnataka, India.

Abstract—Virtualized cloud-based services can take advantage of statistical multiplexing across applications to yield significant cost savings to the operator. In this paper, we seek to lower a provider’s costs of real-time IPTV services through a virtualized IPTV architecture and through intelligent time shifting of service delivery. We take advantage of the differences in the deadlines associated with Live TV versus Video-on-Demand (VoD) to effectively multiplex these services.

We provide a generalized framework for computing the amount of resources needed to support multiple services, without missing the deadline for any service. We construct the problem as an optimization formulation that uses a generic cost function. We consider multiple forms for the cost function (e.g., maximum, convex and concave functions) to reflect the different pricing options. The solution to this formulation gives the number of servers needed at different time instants to support these services. We implement a simple mechanism for time-shifting scheduled jobs in a simulator and study the reduction in server load using real traces from an operational IPTV network.

Keywords: Cloud computing, IPTV, Live TV, Video-on-Demand, optimization, earliest deadline first, server-capacity region.

I. INTRODUCTION

IP-based video delivery becomes more popular, the demands placed upon the service provider’s resources have dramatically increased. Service providers typically provision for the peak demands of each service across the subscriber population. However, provisioning for peak demands leaves resources under utilized at all other periods. This is particularly evident with Instant Channel Change (ICC) requests in IPTV.

Our goal is to take advantage of the difference in workloads of the different IPTV services to better utilize the deployed servers. In IPTV, Live TV is typically multicast from servers using IP Multicast, with one group per TV channel. Video-on-Demand (VoD) is also supported by the service provider, with each request being served by a server using a unicast stream. For each channel change, the user has to join the multicast group associated with the channel, and wait for enough data to be buffered before the video is displayed; this can take some time. As a result, there have been many attempts to support instant channel change by mitigating the user perceived channel switching latency [1], [7]. In our virtualized environment, ICC is typically managed by a set of VMs while other VMs would be created to handle VoD requests. With the ability to spawn VMs quickly [1], we believe that we can shift servers (VMs) from VoD to handle the ICC demand in a matter of a few seconds. This requires being able to predict the ICC bursts which we believe can be predicted from historic information.

Our goal is to find the number of servers that are needed at each time instant by minimizing a cost function while at the same time satisfying all the deadlines associated with these services. To achieve this, we identify the server-capacity region formed by servers at each time instant such that all the arriving requests meet their deadlines. We show that for any server tuple with integer entries inside the server-capacity region, an earliest deadline first (EDF) strategy can be used to serve all requests without missing their deadlines. This is an extension of previous result where the number of servers is fixed [2].

Thus, well known concave programming techniques without integer constraints can be used to solve the problem [3]. Finally, for a maximum cost function, we seek to minimize the maximum number of servers used over the entire period.

II. RELATED WORK

There are mainly three threads of related work, namely cloud computing, scheduling with deadline constraints, and optimization. Cloud computing has recently changed the landscape of Internet based computing, whereby a shared pool of configurable computing resources (networks, servers, storage) can be rapidly provisioned and released to support multiple services within the same infrastructure [7].

In preliminary work on this topic [4], we analyzed the maximum number of servers that are needed to service jobs with a strict deadline constraint. We also assume non-causal information (i.e., all deadlines are known a priori) of the jobs arriving at each instant. In this [5], considers the advancing scenario, this approach only requires a server complex that is sized to meet the requirements of the ICC load, which has no deadline flexibility, and we can almost completely mask the need for any additional servers for dealing with the VoD load. With the typical ICC
implemented on current IPTV systems, the content is delivered at an accelerated rate using a unicast stream from the server [6], [7]. There have been multiple efforts in the past to analytically estimate the resource requirements for serving arriving requests which have a delay constraint. These have been studied especially in the context of voice, including delivering VoIP packets, and have generally assumed the arrival process is Poisson [8]. For a concave minimization with linear constraints, the solution is one of the corner points of the polytope formed by the linear constraints [9], [10].

III. IMPROVED CLOUD RESOURCE UTILIZATION FOR IPTV TRANSMISSION

Internet Protocol-based video delivery is increasing in popularity with the result that its resource requirements are continuously growing. It is estimated that by the year 2017 video traffic will account 69% of the total consumer’s Internet traffic. Content and service providers typically configure their resources such that they can handle peak demands of each service they provide across the subscriber population.

The solution presented takes advantage of the temporal differences in the demands from these IPTV workloads to better utilize the servers that were deployed to support these services. While VoD is delivered via unicast, Live TV is delivered over multicast to reduce bandwidth demands. However, to support Instant Channel Change (ICC) in Live TV, service providers send a unicast stream for that channel for a short period of time to keep a good quality of experience. If a number of users change their channels around the same period of time, this produces a large burst load on the server that has to support the corresponding number of users. Compared to the ICC workload which is very bursty and has a large peak to average ratio, VoD has a relatively steady load and imposes a relatively lax delay requirement. By multiplexing across these services, the resource requirements for supporting the combined set of services can be reduced.

Two services that have workloads which differ significantly over time can be combined on the same virtualized platform. This allows for scaling of the number of resources according to each service’s current workloads. It is, however, possible that the peak workload of different services may overlap. Under such scenarios, the benefit of a virtualized infrastructure diminishes, unless there is an opportunity to time shift one of the services in anticipation of the other service’s requirements to avoid having to deliver both services at the same time instant. In general, the cloud service provider strives to optimize the cost for all time instants, not necessarily just reducing the peak server load.

A. Cost Function

We investigate linear, convex, and concave functions (See Fig. 1). With convex functions, the cost increases slowly initially and subsequently grows faster. For concave functions, the cost increases quickly initially and then flattens out, indicating a point of diminishing unit costs (e.g., slab or tiered pricing). Minimizing a convex cost function results in averaging the number of servers (i.e., the tendency is to service requests equally throughout their deadlines so as to smooth out the requirements of the number of servers needed to serve all the requests). Minimizing a concave cost function results in finding the extremal points away from the maximum to reduce cost. This may result in the system holding back the requests until just prior to their deadline and serving them in a burst, to get the benefit of a lower unit cost because of the concave cost function (e.g., slab pricing). The concave optimization problem is thus optimally solved by finding boundary points in the server-capacity region of the solution space.

We consider the following cost functions:

1) Linear Cost: \( C(s_1, s_2, \ldots, s_T) = \sum_{i=1}^{T} s_i \) This models the case where we incur a cost that is proportional to the total number of servers needed across all times.

2) Convex Separable Cost: \( C(s_1, s_2, \ldots, s_T) = \sum_{i=1}^{T} C(s_i) \) where \( C(s_i) \) is a convex function. This models the case when a data center sees an increasing per unit cost as the number of servers required grows. We consider two examples of \( C(s_i) \), the component cost function. The first is the exponential function, \( C(s_i) = \exp(s_i) \). The second is a piecewise linear function of the form \( C(s_i) = s_i + c(s_i - K) \) where \( c, K \geq 0 \). This component cost function has per-server cost of unity when \( s_i \leq K \), and per-server cost of \( 1 + c \) thereafter.

3) Concave Separable Cost: \( C(s_1, s_2, \ldots, s_T) = \sum_{i=1}^{T} C(s_i) \) with component cost \( C(s_i) \) a concave function. This may arise when the per-server cost diminishes as the number of servers grows.

4) Maximum Cost: \( C(s_1, s_2, \ldots, s_T) = \max_{i=1}^{T} s_i \) This cost function penalizes the peak capacity that will be needed to serve the incoming sequence of requests.

B. Optimal Solutions

1) Linear Cost Function: One strategy for meeting this cost is to set \( s_i = \sum_{j=1}^{K} r(i) \), which means that we serve all requests as they arrive.

2) Piecewise Linear Convex Cost Function: Consider any scheduling of the incoming requests which uses \( y_i \) server resources a time \( i \). Suppose that we only serve \( \min(y_i, K) \) of the requests and drop the remaining. The cost of using servers \( y_i \) at time \( i \) is given by total number of requests \( + c \) times the number of dropped requests. Suppose that when
we use earliest deadline first as the strategy with \( K \) as the number of servers, \( \delta_i \) be the number of requests served in time \( i \) and \( \delta_i \) is the number of requests dropped (Note that \( \delta_i = 0 \) if \( \delta_i < K \)). Then \( s_i = \delta_i + \delta_i \) is an optimal solution.

3) Two-Tiered Step Cost Function: This cost function is neither convex, nor concave. At any time \( i \), suppose that there are \( l_i \) requests that are pending and have deadline \( i \) and a total of \( y_i \) requests that are pending and have not been serviced yet. If \( l_i < n_1 \), \( \min(n_1,l_i) \) requests are served using \( \min(n_1,l_i) \) servers. However, if \( n_1 \leq l_i \leq n_2 \), \( \min(n_2,y_i) \) requests are served (in order of deadline) using \( \min(n_2,y_i) \) servers. We note that \( l_i \leq n_2 \) and hence never is infinite cost incurred.

4) Exponential Cost Function: This is a convex optimization problem with integer constraints, and is thus NP hard problem in general. We here provide an achievable solution based on convex primal-dual method.

\[
\text{Initialization: Let all } s_i = \sum_{j=1}^{k} r_j(i).
\]

Repeat \((n = 1, \ldots,)\):

1. \( \lambda_{i+1} = (\lambda_i + \delta_i(P(i, i_2) - \sum_{j=1}^{y_i} s_j))^+ \), where \( \delta_i = 1/\sqrt{\lambda_{i+1}} \) for all \( 1 \leq i_1 \leq i_2 \leq T \).
2. \( s_i = (\sum_{i=1}^{T} \lambda_i) \) for all \( i = 1, \ldots, T \).
3. \( s_i = s_i + \max_{i \\leq j \\leq T} (P(i, i_2) - \sum_{j=1}^{y_i} s_j)^+ \) for all \( i = 1, \ldots, T \).
4) Concave Cost Function: This is a concave programming with linear and integer constraints. For a concave minimization with linear constraints, the solution is one of the corner points of the polytope formed by the linear constraints [9]. We will show that minimization with linear and an integer constraint is the same as minimization with just the linear constraints. Thus, the solution to the problem is one of the corner points of the polytope formed by the linear constraints.

5) Maximum Cost Function: For this cost function, Suppose that there are arrival processes \( r_j(i) \) for \( 1 \leq j \leq k \) and \( 1 \leq i \leq T \) to a queue at time \( i \). Request \( r_j(i) \) arriving at time \( i \) has a deadline of \( \min(i+d_j) \).

IV. EXISTING SYSTEM

A. Traditional Data Collection

Servers in the VHO serve VoD using unicast, while Live TV is typically multicast from servers using IP Multicast. When users change channels while watching live TV, we need to provide additional functionality so that the channel change takes effect quickly. For each channel change, the user has to join the multicast group associated with the channel, and wait for enough data to be buffered before the video is displayed; this can take some time. As a result, there have been many attempts to support instant channel change by mitigating the user perceived channel switching latency.

B. Disadvantages

1) More Waiting Time
2) More Switching latency
3) Not Cost effective

V. PROPOSED SYSTEM

We propose a) To use a cloud computing infrastructure with virtualization to handle the combined workload of multiple services flexibly and dynamically, b) To either advance or delay one service when we anticipate a change in the workload of another service, and c) To provide a general optimization framework for computing the amount of resources to support multiple services without missing the deadline for any service.

Our goal in this paper is to take advantage of the difference in workloads of the different IPTV services to better utilize the deployed servers. For example, while ICC workload is very bursty with a large peak to average ratio, VoD has a relatively steady load and imposes “not so stringent” delay bounds. A typical service provider network infrastructure layout, an end to end logical architecture is shown in Fig. 2. At the top of the hierarchy is the Super Head End Office (SHO) where both linear programming broadcast content and VoD content are acquired. Content acquired from the SHO is typically carried over an IP backbone network to each of the Video-Hub-Offices (VHO). The content goes to each home from the VHO via the metro-area network into each user’s home and to their set-top box. With the typical ICC implemented on current IPTV systems, the content is delivered at an accelerated rate using a unicast stream from the server [6], [7]. The playout buffer is filled quickly, and thus keeps switching latency small. Once the playout buffer is filled up to the playout point, the set top box reverts back to receiving the multicast stream for the new channel. In the current architecture, this demand is served by a large number of servers that are scaled up as the number of subscribers increases. However this demand is transient and typically only lasts a few seconds (15–60 secs.) As a result, a majority of the servers dedicated to ICC sit idle outside the burst period. Since the servers for ICC are different from the VoD servers, the number of servers scale as the sum of peak requirements of the two services.
A. Advantages
In this paper, we consider two potential strategies for serving VoD requests. The first strategy is a postponement based strategy. In this strategy, we assume that each chunk for VoD has a deadline seconds after the request for that chunk. This would let the user play the content up to seconds after the request. The second strategy is an advancement based strategy. In this strategy, we assume that requests for all chunks in the VoD content are made when the user requests the content. Since all chunks are requested at the start, the deadline for each chunk is different with the first chunk having deadline of zero, the second chunk having deadline of one and so on. With this request pattern, the server can potentially deliver huge amount of content for the user in the same time instant violating downlink bandwidth constraint.

B. Modules
1) Optimization Framework
2) Impact of Cost Functions on Server Requirements
3) Linear Cost Function

Modules Description:
1) Optimizing Framework:
An IPTV service provider is typically involved in delivering multiple real time services, such as Live TV, VoD and in some cases, a network-based DVR service. Each unit of data in a service has a deadline for delivery. For instance, each chunk of video file for VoD need to be serviced by its playback deadline so that the playout buffer at the client does not under-run. In this section, we analyze the amount of resources required when multiple real time services with deadlines are deployed in a cloud infrastructure. There have been multiple efforts in the past to analytically estimate the resource requirements for serving arriving requests which have a delay constraint. These have been studied especially in the context of voice, including delivering VoIP packets, and have generally assumed the arrival process is Poisson.

2) Impact of Cost Function on Server Requirements :
We investigate linear, convex, and concave functions. With convex functions, the cost increases slowly initially and subsequently grows faster. For concave functions, the cost increases quickly initially and then flattens out, indicating a point of diminishing unit costs (e.g., slab or tiered pricing). Minimizing a convex cost function results in the optimal amount of resource (i.e., number of servers at different times) for accommodating multiple services with different deadlines. The initial theoretical framework depends on non-causal information regarding the arrival times and deadlines for each chunk of a requested content. We demonstrate two optimization approaches namely, postponing and advancing VoD delivery. Alternatively, VoD requests can also be advanced after the initial movie request without incurring any startup delays (i.e., subsequent chunks of the movie can be advanced before their playout deadlines).

A. Maximum Cost Function, Postponement Scenario
We set up a series of experiments to see the effect of varying firstly, the ICC durations and secondly, the VoD delay tolerance on the total number of concurrent streams needed to accommodate the combined workload. In figures diurnal VoD time series (in blue) and a ICC time series (in red). For a given VoD Delay $n \geq 0$, we use two services, one with delay 0 and one with delay $n$. For each incoming VoD movie request without incurring any startup delays (i.e., subsequent chunks of the movie can be advanced before their playout deadlines).

The legends in each plot indicate the duration that each VoD session can be delayed by ($n$). Fig. 4 shows the superposition of the number of VoD sessions with a periodic synthetic LiveTV ICC session. The duration of the ICC burst is set to be 15 seconds and the peak of the pulse is set to be the peak of the VoD concurrent streams for the whole day. We now compute the total number of concurrent streams that are necessary and sufficient to serve all the incoming requests.

$$S = \max\{\sum_{j=1}^{N} \sum_{i=0}^{\min(n, D_{t})} \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} r_{i}(n) \}$$

The legends in each plot indicate the duration that each VoD session can be delayed by ($n$). Fig. 4 shows the superposition of the number of VoD sessions with a periodic synthetic LiveTV ICC session. The duration of the ICC burst is set to be 15 seconds and the peak of the pulse is set to be the peak of the VoD concurrent streams for the whole day. We now compute the total number of concurrent streams that are necessary and sufficient to serve all the incoming requests.

$$S = \max\left\{\sum_{j=1}^{N} \sum_{i=0}^{\min(n, D_{t})} \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} r_{i}(n) \right\}$$

The legends in each plot indicate the duration that each VoD session can be delayed by ($n$). Fig. 4 shows the superposition of the number of VoD sessions with a periodic synthetic LiveTV ICC session. The duration of the ICC burst is set to be 15 seconds and the peak of the pulse is set to be the peak of the VoD concurrent streams for the whole day. We now compute the total number of concurrent streams that are necessary and sufficient to serve all the incoming requests.

$$S = \max\left\{\sum_{j=1}^{N} \sum_{i=0}^{\min(n, D_{t})} \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} r_{i}(n) \right\}$$

The legends in each plot indicate the duration that each VoD session can be delayed by ($n$). Fig. 4 shows the superposition of the number of VoD sessions with a periodic synthetic LiveTV ICC session. The duration of the ICC burst is set to be 15 seconds and the peak of the pulse is set to be the peak of the VoD concurrent streams for the whole day. We now compute the total number of concurrent streams that are necessary and sufficient to serve all the incoming requests.

$$S = \max\left\{\sum_{j=1}^{N} \sum_{i=0}^{\min(n, D_{t})} \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} r_{i}(n) \right\}$$

The legends in each plot indicate the duration that each VoD session can be delayed by ($n$).
B. Maximum Cost Function, Advancement Scenario

1) Advancement Subject to No Downlink Bandwidth Constraint:
A movie request is made up of different chunk deadlines. For each chunk, we associate a service class $i$. Specifically, the $i$th chunk of any movie is designated a service class with a corresponding deadline of $i$-1. For a requested movie, we enlist a request made of $L$ service classes (service classes 1 to $L$), where $L$ is the movie length. A LiveTV ICC request corresponds to a service class 1 request for 15 consecutive seconds as in the postponement case.

For an operational trace as shown in Fig. 5, with advancing, a maximum of 24955 concurrent streams can accommodate both LiveTV and VoD requests. With only LiveTV, the total number of concurrent streams needed is 24942. VoD requests can be essentially serviced with just an additional 13 concurrent streams.

2) Advancement Subject to Downlink Bandwidth Constraint:
Practically, the downlink bandwidth of the users is a constraint. Also, there is a limit to the time the chunks can be advanced. We take these two issues into account by putting a maximum constraint on the advancement. Consider a maximum advancement threshold of $n\geq 0$. For a given $n$, we make $\min(n+1,M)$ service classes with deadline of 0 until $\min(n,M-1)$ where $M$ is the maximum over all movie lengths. As soon as the movie request $j$ (with movie length $L$) occurs, a request is made for 1 chunk each of service class 1 to $\min(n+1,L)$, and for all next $(L-n-1)^+$ timeslots, there is a request for a chunk of service class $n+1$. VoD request creates requests of service class 1 in consecutive 15 time-slots as before as in Fig. 6.

C. Convex Piecewise Linear Cost Function

1) Postponement Scenario:
Figs. 7 and 8 show the number of concurrent streams needed as obtained with the optimal strategy when a piecewise linear cost function is used. We note how for different values of $K$ allocates substantially different number of servers. Fig. 7 simulates a synthetic LiveTV ICC 15 sec pulse width of amplitude$\approx 1200$ and an operational VoD measurement trace. When $K=12500$ the number of concurrent streams needed peak with every incoming ICC burst (spaced 30 mins apart) and then bottom out at the VoD envelope (in blue). If we use a smaller $K$, e.g., $K=1000$, many chunks will miss their deadlines (especially during the peak hour) if total of $K$ servers are used. Thus, a large number of chunks have to be served with a higher cost (red spikes of value $2.3x10^4$ during the busy hour). The number of chunks that need to be served with a higher cost is larger when $K$ is smaller. For any $K$ smaller, there would be a chunk that misses deadline with $K$ concurrent streams and hence there will be at-least one chunk that is served at higher cost. Lower the value of $K$, more jobs need to be served with higher cost. Fig. 8 portrays a similar story for an operational LiveTV trace. With a smaller $K$, jobs are delayed to create larger overshoots (red spike of value $3.6x10^4$).
2) Advancement Scenario:
Fig. 9 shows the number of concurrent streams with time by advancing VoD requests with no downlink bandwidth constraint, while Fig. 10 shows the corresponding CDF. Once again, we notice that lower the value of K, more jobs need to be served with higher cost.

Fig. 9. Total number of sessions needed with a 15 sec ICC pulse width, Advancement scenario.

Fig. 10. CDF plot for the total number of of concurrent sessions with a 15 sec ICC pulse width, Advancement scenario.

VIII. CONCLUSION
In this paper, IPTV service providers can leverage a virtualized cloud infrastructure by intelligently time-shifting load to better utilize deployed resources while still meeting the strict time deadlines for each individual service. We used LiveTV ICC and VoD as examples of IPTV services that can run on a shared virtualized infrastructure. Our paper first provided a generalized framework for computing the resources required to support multiple services with deadlines. We formulated the problem as an optimization problem and computed the number of servers required based on a generic cost function. We considered multiple forms for the cost function of the server complex (e.g., min-max, convex and concave) and solved for the optimal number of servers required to support these services without missing any deadlines.

We provide an analysis that computes the minimum number of servers needed to accommodate a combination of IPTV services, namely VoD session and Live TV instant channel change bursts. By anticipating the LiveTV ICC bursts that occur every half hour we can speed up delivery of VoD content by prefilling the set top box buffer. This helps us to dynamically reposition the VoD servers for accommodating the LiveTV bursts that typically last for 15 to 30 seconds at most. Our results show that anticipating and thereby delaying VoD requests gives significant resource savings.

ACKNOWLEDGEMENTS
This work has been partially supported by the AT&T Labs—Research, Florham Park, NJ 07932 USA. A preliminary version of this study has been presented in IEEE IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 15, NO. 4, JUNE 2013. Any view, findings and conclusions expressed in this paper are those of the authors.

REFERENCES