MULTIPLEXING CLOUD RESOURCES FOR DELIVERING IPTV SERVICES

[1]SHAIK MYMOON, [2]CH.SUBBARAO,[3]K.BALA CHOWDAPPA

[1]PG SCHOLAR,CSE(CN), QCET, NELLORE

[2]ASSOCIATE PROFESSOR, CSE, QCET, NELLORE

[3]ASST PROFESSOR, CSE, GPREC, KURNOOL

ABSTRACT: Virtualized cloud-based services can take advantage of statistical multiplexing across applications to yield significant cost savings to the operator. However, achieving similar benefits with real-time services can be a challenge. 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. Our results show that we are able to reduce the load by _ 24% (compared to a possible _ 31%). We also show that there are interesting open problems in designing mechanisms that allow time-shifting of load in such environments.

1.INTRODUCTION: As 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 underutilized at all other periods. This is particularly evident with Instant Channel Change (ICC) requests in IPTV. 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. When users change channels while watching live TV, we need to provide additional functionality to 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. With the typical ICC implemented on IPTV systems, the content is delivered at an accelerated rate using a unicast stream from the server. The play out buffer is filled quickly, and thus keeps switching latency small. Once the play out buffer is filled up to the play ut point, the set top box reverts back to receiving the multicast stream.

ICC adds a demand that is proportional to the number of users concurrently initiating a channel change event. Operational data shows that there is a dramatic burst load placed on servers by correlated channel change requests from consumers. This results in large peaks occurring on every half-hour and hour boundaries and is often significant in terms of both bandwidth and server I/O capacity. Currently, this demand is served by a large number of servers grouped in a data center for serving individual channels, and are scaled up as the number of subscribers increases. However this demand is transient and typically only lasts several seconds, possibly upto a couple of minutes. As a result, majority of the servers dedicated to live TV

sit idle outside the burst period. 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. More importantly, it offers opportunities for the service provider to deliver the VoD content in anticipation and potentially out-of-order, taking advantage of the buffering available at the receivers. We seek to minimize the resource requirements for supporting the service by taking advantage of statistical multiplexing across the different services - in the sense, we seek to satisfy the peak of the sum of the demands of the services, rather than the sum of the peak demand of each service when they are handled independently. Virtualization offers us the ability to share the server resources across these services.

In this paper, we aim a) to use a cloud computing infras-tructure with virtualization to dynamically shift the resources in real time to handle the ICC workload, b) to be able to anticipate the change in the workload ahead of time and preload VoD content on STBs, thereby facilitate the shifting of resources from VoD to ICC during the bursts and c) solve a general cost optimization problem formulation without having to meticulously model each and every parameter setting in a data center to facilitate this resource shift. In a virtualized environment, ICC is managed by a set of VMs (typically, a few VMs will be used to serve a popular channel). Other VMs would be created to handle VoD requests. With the ability to spawn VMs quickly, we believe we can shift servers (VMs) from VoD to handle the ICC demand in a matter of a few seconds. Note that by being able to predict the ICC bursts (channel change behavior can be predicted from historic logs as a result of LiveTV show timings. The channel changes usually occur every half hour. In anticipation of the ICC load, we seek to accelerate delivery of VoD content (for example, for a small number of minutes of play out time) to the users' STBs and shift the VoD demand away from the ICC burst interval . This will also ensure that VoD users will not notice any impairment in their delivered quality of service (e.g. frozen frames etc.) as the play out can be from the local STB cache.

In preliminary work on this topic, we analyzed the maximum number of servers that are needed to service jobs with a strict deadline contraint. We also assume non-causal information (i.e., all deadlines are known a priori) of the jobs arriving at each instant. In this paper, we consider a generalized cost function for the servers. The cost of servers in this model can be a function of time, load, etc. Our goal is to find the number of servers at each time instant by minimizing this generalized cost function while at the same time satisfying all the deadline constraints.

We identify the sever-capacity region formed by servers at each time instant such that all the jobs arriving meet their deadlines, which are defined as: the region such that for any server tuple with integer entries inside this region, all de adlines can be met and for any server tuple with integer entries outside this region, there will be at least one request that misses the deadline. We also 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 results in the literature where the number of servers is fixed at all times. The server-capacity region is formed by linear constraints, and thus this region is a polytope. Having identified the server-capacity region in all its gen-erality, we consider

the cost function to be one of several possibilities: a separable concave function, a separable convex function, or a maximum function. We note that even though the functions are concave/convex, the feasible set of server tuples is all integer tuples in the server-capacity region. This integer constraint makes the problem hard, in general. We show that for a piecewise linear separable convex function, an optimal strategy that minimizes the cost function can be easily described. Furthermore, this strategy only needs causal information of the jobs arriving at each time-instant. For any concave cost function, we show that the integer constraint can be relaxed since all the corner points of the server-capacity region (which is a polytope) have integer coordinates. Thus, well known concave programming techniques without integer constraints can be used to solve the problem. Finally, for a maximum cost function, we seek to minimize the maximum number of servers used over the entire period. This paper finds a closed form expression for the optimal value for the maximum number of servers needed based on the non-causal information of the job arrival process. We show two examples of the cost function for computing the number of servers in Section V namely, the maximum and piecewise linear convex cost functions. We set up a series of numerical simulations to see the effect of varying firstly, the ICC durations and secondly, the VoD delay tolerance on the total number of servers needed to accommodate the combined workload. Our findings indicate that potential server bandwidth savings of (20% - 25%) can be realized by anticipating the ICC load and thereby shifting/smoothing the VoD load ahead of the ICC burst. Finally, we show by means of a faithful simulator implementing both these services in Section VI that a careful choice of a look ahead smoothing window can help to average the additional VoD load. Ultimately our 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.

2.SYSTEM REQUIREMENTS:

2.1. Functional Requirements

Modules:

1. ADMIN
2. USER

Modules Description

ADMIN:

Here admin is a key role player. The admin can add the channel to view users of their requirements. Here admin add channels with domain separation i.e. sports, news, entertainment etc… And also admin can view all users who are using our application.

USER:

Here user is the end-user. User can use our application with a small registration. Once user registered successfully then the user gets some login id and password. By using this login details user can use our application. In our application user can view all the channels which is uploaded by admin.

2.2. Performance Requirements

Performance is measured in terms of the output provided by the application.

Requirement specifications play an important part in the analysis of system. Only when the requirement specifications are properly given, it is

possible to design a system, which will fit into required environment . It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use. The requirement specification for any system can be broadly stated as given below:

- The system should be accurate
- The system should be better than the existing system
- The existing system is completely dependent on the user to perform all the duties.

2.3. Hardware Interface

Processor - Pentium –III

Speed - 1.1 GHz

RAM - 256 MB (min)

Hard Disk - 20 GB

2.4. Software Requirements

OperatingSystem :Windows95/98/2000/XP

Application Server : Tomcat5.0/6.X Front End : HTML, Java, Jsp

Scripts : JavaScript.

Server side Script : Java Server Pages.

Database : Oracle10g

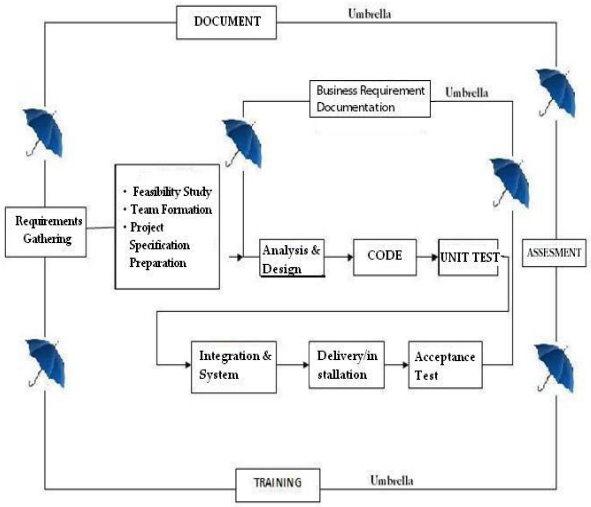
Database Connectivity : JDBC.

2.5. Data Dictionary

A data dictionary is a collection of descriptions of the data objects or items in a data model for the benefit of programmers and others who need to refer to them. A first step in analyzing a system of objects with which users interact is to identify each object and its relationship to other objects. This process is called data modeling and results in a picture of object relationships. After each data object or item is given a descriptive name, its relationship is described (or it becomes part of some structure that implicitly describes relationship), the type of data (such as text or image or binary value) is described, possible predefined values are listed, and a brief textual description is provided. This collection can be organized for reference into a book called a data dictionary.

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

SDLC (Umbrella Model)



Stages of SDLC:

- Requirement Gathering and Analysis
- Designing
- Coding
- Testing
- Deployment

Requirements Definition Stage and Analysis:

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.

HomePage:

Admin Login:

3. CONCLUSION:

We studied how IPTV service providers can leverage a virtualized cloud infrastructure and intelligent time-shifting of load to better utilize deployed resources. Using Instant Channel Change and VoD delivery as examples, we showed that we can take advantage of the difference in workloads of IPTV services to schedule them appropriately on virtualized infrastructures. By anticipating the LiveTV ICC bursts that occur every half hour we can speed up delivery of VoD content before these bursts by profiling the set top box buffer. This helps us to dynamically reposition the VoD servers to accommodate ICC bursts that typically last for a very short time. Our paper provided generalized framework for computing the amount of resources needed to support multiple services with deadlines. We formulated the problem as a general optimization problem and computed the number of servers required according to a generic cost function. We considerd multiple forms for the cost function (e.g., min-max, convex and concave) and solved for the optimal number

of servers that are required to support these services without missing any deadlines. We implemented a simple time-shifting strategy and eval-uated it using traces from an operational system. Our results show that anticipating ICC bursts and time-shifting VoD load gives significant resource savings (as much as 24%). We also studied the different parameters that affect the result and show that their ideal values vary over time and depend on the relative load of each service. mechanisms as part of our future work.

REFERENCES

[1] D. Banodkar, K. K. Ramakrishnan, S. Kalyanaraman, A. Ger ber, and O. Spatscheck, "Multicast instant channel change in IPTV system," in Proceedings of IEEE COMSWARE, January 2008.

[2]"Microsoftv:Iptvedition,"http://www.microsoft.com/tv/IPTVEditio n.mspx.

[3] H. A. Lagar-Cavilla, J. A. Whitney, A. Scannell, R. B. P. Patchin, S. M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan, "SnowFlock: Virtual Machine Cloning as a First Class Cloud Primitive," ACM Transactions on Computer Systems (TOCS), 2011.

[4] V. Aggarwal, V. Gopalakrishnan, R. Jana, K. K. Ramakrishnan, and V. Vaishampayan, "Exploiting Virtualization for Delivering Cloud-based IPTV Services," in Proc. of IEEE INFOCOM (mini-conference), Shang-hai, April 2011.

[5] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo, Dead-line Scheduling for Real-Time Systems: Edf and Related Algorithms. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[6] N. V. Thoai and H. Tuy, "Convergent algorithms for minimizing a concave function," in Mathematics of operations Research, vol. 5, 1980.

[7] R. Urgaonkar, U. Kozat, K. Igarashi, and M. J. Neely, "Dynamic resource allocation and power management in virtualized data centers," in Proceedings of IEEE IFIP NOMS, March 2010.

[8] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogram-ming in a Hard Real Time Environment," Journal of the ACM, vol. 20, no. 1, pp. 46–61, 1973.