The HKU Scholars Hub The University of Hong Kong 香港大學學術庫

Title	CloudMedia: When cloud on demand meets video on demand
Author(s)	Wu, Y; Wu, C; Li, B; Qiu, X; Lau, FCM
Citation	The 31st International Conference on Distributed Computing Systems (ICDCS 2011), Minneapolis, MN., 20-24 June 2011. In Proceedings of 31st ICDCS, 2011, p. 268-277
Issued Date	2011
URL	http://hdl.handle.net/10722/135699
Rights	Creative Commons: Attribution 3.0 Hong Kong License

2011 31st International Conference on Distributed Computing Systems

CloudMedia: When Cloud on Demand Meets Video on Demand

Yu Wu*, Chuan Wu*, Bo Li†, Xuanjia Qiu*, Francis C.M. Lau*
*Department of Computer Science, The University of Hong Kong, Email: {ywu,cwu,xjqiu,fcmlau}@cs.hku.hk
†Department of Computer Science and Engineering, Hong Kong University of Science and Technology,
Email: bli@cse.ust.hk

***Abstract*—Internet-based cloud computing is a new computing paradigm aiming to provide agile and scalable resource access in a utility-like fashion. Other than being an ideal platform for computation-intensive tasks, clouds are believed to be also suitable to support large-scale applications with periods of flash crowds by providing elastic amounts of bandwidth and other resources on the fly. The fundamental question is how to configure the cloud utility to meet the highly dynamic demands of such applications at a modest cost. In this paper, we address this practical issue with solid theoretical analysis and efficient algorithm design using Video on Demand (VoD) as the example application. Having intensive bandwidth and storage demands in real time, VoD applications are purportedly ideal candidates to be supported on a cloud platform, where the on-demand resource supply of the cloud meets the dynamic demands of the VoD applications. We introduce a queueing network based model to characterize the viewing behaviors of users in a multi-channel VoD application, and derive the server capacities needed to support smooth playback in the channels for two popular streaming models: client-server and P2P. We then propose a dynamic cloud resource provisioning algorithm which, using the derived capacities and instantaneous network statistics as inputs, can effectively support VoD streaming with low cloud utilization cost. Our analysis and algorithm design are verified and extensively evaluated using large-scale experiments under dynamic realistic settings on a home-built cloud platform.**

I. Introduction

Cloud computing has recently emerged as a new computing paradigm for organizing a shared pool of servers in datacenters into a *cloud* infrastructure that can provide on-demand server utilities (CPU, storage, bandwidth, etc.) to users anywhere anytime. To enable different applications running on a cloud efficiently, virtualization is often applied, which allows multiple virtual machines (VMs) to run on the same physical server; this form of sharing a physical server allows resources to be rapidly provisioned and released with minimal management efforts and overheads. Resource provisioning is typically based on Service Level Agreements (SLAs) between the cloud provider and the cloud consumer. Different service models of a cloud infrastructure have been proposed ([1]), namely Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), among which the IaaS model provides the most flexibility where a consumer can deploy and run any software on its allocated VMs.

The research was supported in part by grants from RGC under the contracts 718710E and 615608, and by a grant from Huawei Technologies Co. Ltd. under the contract HUAW18-15L0181011/PN.

The elastic and on-demand nature of resource provisioning has made cloud computing very promising in various applications, including many that are computation-intensive [2], [3] and applications with highly dynamic server resource demands [4]. In particular, dynamic resource provisioning via a cloud is best suited for Internet-based applications (*e.g.* YouTube) that have to handle frequent surges of user requests. Real-life cloud implementations have demonstrated that cloud infrastructures indeed have substantial advantages over private server clusters or CDNs in terms of system scalability, and they can lead to significant reduction in operational costs with respect to machines, bandwidth, and management.

Our work focuses on Internet-based applications that are to be supported by cloud infrastructures. Little effort so far has been devoted to understanding and exploring how these Internet-based applications can fully exploit a cloud infrastructure. One fundamental question is how to quantify dynamic user demands, or more precisely, *how should an application provider learn the dynamic demands from users and relay them to the cloud service accordingly?* Our work as presented in this paper makes the first attempt to address this problem. Specifically, our solution answers the question of how an application provider can most effectively configure the cloud utility to achieve the best application performance at a reduced cost. We choose Video on Demand (VoD) as the representative application in this study.

With intensive and dynamic bandwidth/storage demands in real time, VoD applications present a significant challenge to resource provisioning in service offering. Although many popular VoD services (*e.g.*, PPLive, UUSee) have leveraged peer-to-peer (P2P) technology for cost reduction, existing studies showed that dedicated servers are still catering for $40-70\%$ of overall streaming bandwidth demand in these systems [5], [6]. The cloud infrastructure as an alternative to dedicated servers sets out to meet the challenge by dynamically composing and optimizing the needed services at reduced costs. Our contributions in this study are as follows.

First, we investigate by solid theoretical analysis the equilibrium demand for streaming server capacity in a VoD application with multiple video channels. We introduce a new queueing network model to characterize the dynamic viewing behaviors of VoD users, and derive the server capacity needed to support smooth playback in the channels. Both client-server VoD and P2P VoD are investigated, which are the two most

1063-6927/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDCS.2011.50

IEEE computer society

representative implementations of VoD applications in today's Internet.

Second, by following practical cloud charging models, we formulate two optimization problems for cloud provisioning, including VM provisioning and storage rental, in a cloud infrastructure. We propose a practical algorithm, which can dynamically configure cloud resources to address the continuous demands for streaming different chunks and videos over time. With this algorithm and using instantaneous network statistics as inputs, a VoD application provider periodically derives the required server resources by approximately solving the two optimization problems using efficient heuristics, and communicates the results to the cloud provider using SLAs.

Third, based on a VoD prototype running on a cloud platform (called *CloudMedia*) we have built and implemented using a cluster of machines, we carry out verification of our analysis and extensive evaluation of our algorithm design in a dynamic and realistic environment. The results show that high-performance multi-channel VoD streaming can be implemented with low cloud (server) costs using our algorithm, and that by engaging a cloud in lieu of dedicated servers, a P2P VoD application can readily enjoy the benefits of further reduced costs and improved scalability.

The remainder of this paper is organized as follows. We discuss related work in Sec. II. In Sec. III, we present the model of the cloud infrastructure, as well as that of the multi-channel VoD application. We introduce a new Jackson queueing network model and derive server capacity demand in the VoD system in Sec. IV. We formulate the cloud VM provisioning and storage rental optimization problems, and propose a practical dynamic cloud provisioning algorithm in Sec. V. Sec. VI presents our extensive experimental evaluations under realistic settings. Finally, we conclude the paper in Sec. VII.

II. Related Work

Recently there is an upsurge of interest in the research community in issues arising from running computation-intensive and data-intensive applications on clouds [3][7][8][9][10][11]. Many of these applications can now be satisfactorily supported by commercial cloud services [12][13]. Researchers however have focused mainly on how a cloud infrastructure can provide for the quality of service (QoS) as required by the application and based on the SLA negotiated [7][8], and how user privacy and content confidentiality can be protected when a user entrusts a cloud with a task [9][10]. Our work deviates from these popular topics and focuses instead on the issue of cloud's utilization by large-scale Internet-based applications. Our study is from the viewpoint of a cloud consumer, *i.e.*, an Internet application provider in our case. It seems that no existing work has ever addressed the challenges or proposed any approach from such an angle.

There have been many theoretical studies on performance modeling of P2P streaming applications, *e.g.*, on maximum sustainable streaming rate or smooth streaming probabilities in P2P live streaming [14][15][16], and on start-up delay performance in P2P VoD streaming [17]. These work typically assume a fixed server capacity in the analysis, and none has conducted their study from the perspective of equilibrium server capacity which is needed to maintain a set level of user playback performance in dynamic VoD networks. Wu *et.al* [15] leverage Jackson queueing network with infinite-server queues to model the channel churns, which might not apply to real systems due to the impractical assumption that server resources are unlimited. We are only aware of one study [18] which discusses the minimum server bandwidth required to support a fixed streaming rate in P2P live streaming for set top boxes without any dynamics. However, their bounds are achieved via a tree-based algorithm and are most likely not applicable to real-world VoD streaming service. To the best of our knowledge, our study is the first to put forward a Jackson queueing network model to derive the demand for server capacity in Internet-based P2P VoD streaming which is much more challenging than live streaming due to aggravated chunk availability issues and user dynamics.

As for practical server capacity provisioning, existing studies either focus on provisioning dedicated private servers by the application provider [19][20], or scheduling server resources among multiple applications inside a cloud by a cloud provider [11]. In this paper, we seek to design a resource provisioning framework from the viewpoint of a cloud consumer.

III. System Models

A. The Cloud Infrastructure

The IaaS cloud system under our investigation consists of a collection of interconnected computing and storage servers. The servers are of two categories: *NFS storage servers*, organized into a number of NFS clusters by their performance levels (*e.g.*, storage capacity and I/O speeds), and *computing servers* which support the running and provisioning of *virtual machines* (VMs). The VM instances generally have different configuration levels in terms of CPU computing units and I/O speeds. There are a number of *virtual clusters*, each consisting of VMs of the same level of configuration. Cloud applications run on the VMs and utilize the NFS storage system via the VMs. We assume each VM can access all the NFS clusters via high-speed Ethernet switches and LAN buses.

The cloud architecture is illustrated in Fig. 1, with the following main functional modules:

- ▷ **Broker** is a communicating interface between the cloud provider and a cloud consumer, via which the consumer can submit requests to the cloud.
- ▷ **Request Monitor** listens to the requests from the consumers (brokers), and forwards them to the SLA negotiator.
- ▷ **SLA Negotiator** negotiates the Service Level Agreements (SLAs) with the cloud consumers based on the pricing policy and QoS levels set by the cloud provider.
- ▷ **VM Scheduler** is responsible for VM provisioning to meet the demands of the applications.
- ▷ **VM Monitor** keeps track of all the VM instances provisioned and monitors their activities and performance.

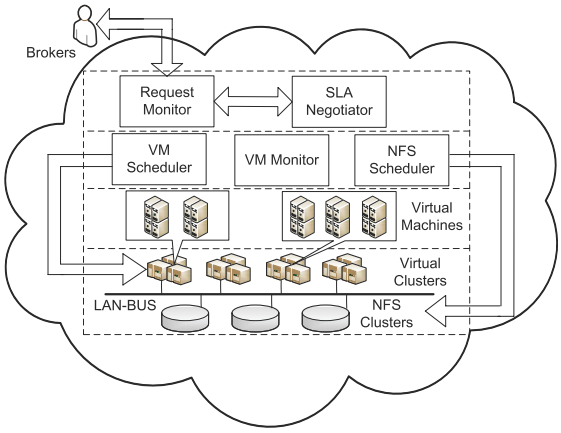

Fig. 1. The cloud model.

▷ **NFS scheduler** carries out content placement onto the NFS clusters.

In this cloud model, services are charged by usage time, following the charging model of leading commercial cloud providers such as Amazon EC2 [12] and S3[13]. Two types of charges are levied on a cloud consumer, the rental fees of the VMs to run the application and the storage cost to use the NFS clusters, both of which are based on a per time unit rate, with different VMs (NFS servers) of different configuration levels requesting different prices.

B. The VoD Application Model

As a cloud consumer, the VoD application possesses a large collection of videos (each referred to as a *video channel*). A user can join any of the channels and watch any portion of a video at any given time. Suppose the streaming playback rate of each channel is r bytes/s, and each video stream is divided into consecutive chunks of size rT_0 bytes, corresponding to T_0 seconds of the playback at rate r. The size of the local playback buffer at each user is sufficient to cache any one video in the system, and a chunk of a channel once downloaded will remain available in the buffer until the user leaves the channel.

We study two models of VoD implementation in this paper, *i.e.*, the client-server model and the P2P paradigm. In the client-server model, all users directly obtain the chunks of their desired videos from the streaming servers, *i.e.*, the cloud infrastructure in our design. For the P2P VoD model, we assume a mesh-pull based P2P VoD design, where users watching the same video channel are organized into a mesh overlay, exchange video chunks among themselves based on periodically exchanged buffer availability bitmaps, and resort to streaming servers only when deemed necessary. In state-of-the-art P2P VoD systems [21], streaming servers are still largely indispensable as they are the only persistent sources of all original videos and needed from time to time to compensate for insufficient upload bandwidth of some peers. In our design, such streaming server service will be implemented by the cloud service.

TABLE I
NOTATION TABLE

Symbol	Definition
r	The streaming playback rate of each video channel
R	The allocated bandwidth of each VM
T_0	The playback time of a video chunk
$J^{(c)}$	Number of chunks that channel c is divided into
$Q_i^{(c)}$	The i-th queue in channel c
$s_i^{(c)}$	The upload bandwidth to serve chunk i in channel c
$m_i^{(c)}$	The number of (queueing theoretical) servers in queue $Q_i^{(c)}$
$\Delta_i^{(c)}$	The capacity provisioned from the cloud for chunk i in channel c
$P_{ij}^{(c)}$	The probability that users in $Q_i^{(c)}$ switch to $Q_j^{(c)}$
$\Lambda^{(c)}$	The external arrival rate to channel c
$\lambda_i^{(c)}$	The aggregate arrival rate to $Q_i^{(c)}$ in channel c
μ	The service rate of each server in a multiple-servers queue
α	The fraction of peers who enter the first chunk queue upon joining a channel
$p_i^{(c)}(k)$	The probability that $Q_i^{(c)}$ has k peers.
$n_i^{(c)}$	The number of peers in $Q_i^{(c)}$
$\nu_i^{(c)}$	The total number of chunk i of channel c in the P2P overlay (equivalently the number of peers in the overlay who own chunk i)
$\nu_{ij}^{(c)}$	The number of peers in $Q_j^{(c)}$ who own chunk i
$\Gamma_i^{(c)}$	The capacity contributed to upload chunk i in channel c from the P2P overlay

C. Interplay between Cloud and VoD Application

As a cloud consumer, the VoD application provider places the video contents onto the NFS clusters and deploys the VoD server application onto the VMs, eliminating the need for traditional streaming servers. To support such a content distribution application, each VM in the cloud is assumed to be assigned a guaranteed amount of bandwidth based on QoS provisioning; the bandwidth provisioned to each VM is R, which is the same for all VMs and satisfies $R > r$, without loss of generality. Over time, the VoD application provider requests different numbers of VMs and different amounts of storage capacity from the cloud via the broker. The requests are based on the current demand from the VoD users, as well as the operational budget and the SLA with the cloud provider. The cloud processes the requests received via the request monitor and adjusts VM and NFS storage allocations via the VM and NFS schedulers.

Our objective in this paper is to study how a VoD application provider can meticulously configure its usage of the cloud infrastructure to achieve the best performance of the application with a modest cost over time. Issues regarding the implementation of the function modules and the QoS provisioning (*e.g.*, allocating bandwidths to the VMs) in the cloud infrastructure represent orthogonal research problems, which are out of the scope of the current paper.

We summarize important notations used in the paper in Table I for ease of reference. We refer to the VoD system using the cloud infrastructure to implement streaming servers as *CloudMedia* hereinafter.

IV. SERVER CAPACITY DEMAND ANALYSIS: A QUEUEING NETWORK APPROACH

We first study the equilibrium demand for streaming server capacity in a VoD application in both the client-server and the

P2P mode, the results of which can translate into guidelines for the VoD provider to configure its cloud usage.

A. Queueing Network Modeling

We introduce a Jackson queueing network based model to characterize the viewing behaviors of VoD users inside each channel in the VoD system. The model facilitates our study of the server capacity needed to support smooth playback in the channels.

A *Jackson Network* [22] is a network of queues where the arrivals at each queue form a Poisson process, and the job service times are exponentially distributed. An *open* Jackson Network is one with external job arrivals into or departures from the system. For each video channel c, let $J^{(c)}$ be the number of chunks the video is divided into. The viewing behavior of users in channel c can be modeled as an open Jackson queueing network.

We model each chunk i in channel c as a queue $Q_i^{(c)}, i = 1, \ldots, J^{(c)}$. A user downloading a chunk is a job in the corresponding queue. The user arrival to download a chunk equals the job arrival at the queue, and finishing downloading a chunk maps to completing a job in the queueing. The queueing time of a job in the queue corresponds to the waiting time of the user for available bandwidth for the download. The service time of a job in a queue maps to the actual chunk download time of a user.

There are $m_i^{(c)}$ servers[1] in queue $Q_i^{(c)}$ with service rate μ each, and the service time of a job in a server is assumed to be exponentially distributed with an average of $\frac{1}{\mu}$. The service rate μ of each server maps to the bandwidth R of each VM in the cloud infrastructure (to simplify later computation of the number of VMs serving each chunk) with $R = \mu \times rT_0$ (recall rT_0 is the size of each chunk in bytes). As assumed in the previous section, R should be larger than the streaming playback rate r, to make it possible that the retrieval of a chunk (of playback time T_0) can be completed within time T_0 considering both waiting and actual download times. The total service rate of $m_i^{(c)}$ servers in queue $Q_i^{(c)}$ maps to the overall available bandwidth to upload chunk i in the network, which is $s_i^{(c)} = \mu m_i^{(c)} \times rT_0 = Rm_i^{(c)}$. The *sojourn* time of a job, *i.e.*, the sum of queueing and service times, corresponds to the total time a user spends on the retrieval of a chunk.

Let $\mathbf{P}^{(c)}$ denote the chunk transfer probability matrix of channel c, with entries $P_{ij}^{(c)}$ representing the probability that a user downloading chunk i will move on to download chunk j (j may or may not be consecutive to i). The transfer probabilities reflect viewing behaviors of the VoD users. They satisfy $\sum_{j=1}^{J^{(c)}} P_{ij}^{(c)} \leq 1, \forall i$, and $1 - \sum_{j=1}^{J^{(c)}} P_{ij}^{(c)}$ is the probability that a user downloading chunk i leaves the channel. Correspondingly, all queues in the Jackson network are interconnected, with job transition probability indicated by $\mathbf{P}^{(c)}$.

External user arrival into the channel and departure from the channel map to the external arrival and departure in the open Jackson network. Without loss of generality, we assume a fraction α of the arrived users start watching the channel from the beginning, *i.e.*, they go into queue $Q_1^{(c)}$, and the rest $1-\alpha$ of the arrived users choose to start with the other chunks with a uniform probability.

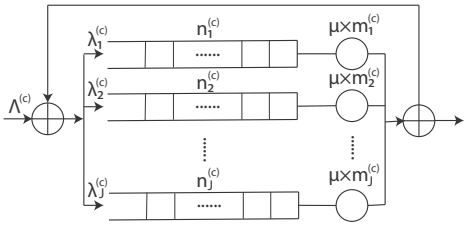

Fig. 2. Channel-level queueing network model.

We assume the external arrival of users to the channel follows a Poisson process with an average arrival rate of $\Lambda^{(c)}$.[2] Together with the assumption that the service time in each chunk queue is exponentially distributed, we can derive that the arrival to each queue is Poisson, since sub-flows resulting from stochastically splitting a Poisson flow are still Poisson, and the aggregation of multiple Poisson flows is still a Poisson flow. Therefore, the queueing network we have modeled is an open Jackson queueing network with a number $J^{(c)}$ of $M/M/m_i^{(c)}/\infty$ queues. An illustration of the queueing network is given in Fig. 2, where $\lambda_i^{(c)}$ is the arrival rate to queue $Q_i^{(c)}$, and $n_i^{(c)}$ is the number of users currently in the queue (including both waiting and downloading ones), $\forall i = 1, \ldots, J^{(c)}$.

B. Client-Server VoD

Based on the queueing network model, we now study the amount of upload capacity needed to support smooth playback in each channel. We first consider the case that the VoD application is implemented in the client-server mode.

We study the upload capacity needed to serve each chunk in channel c, such that a smooth playback can be achieved at the users in the stable state of the VoD system. Mapping it into the Jackson network, we derive the required number of servers $m_i^{(c)}$ of each queue $Q_i^{(c)}, i = 1, \ldots, J_i^{(c)}$, in the equilibrium state, such that the expected sojourn time of each user in each chunk queue is T_0. Recall that T_0 is the playback time of each chunk at the streaming playback rate r in the VoD system. To guarantee smooth playback at each user, the average time of retrieving each chunk (waiting plus downloading) should be no longer than the playback time of the chunk. By setting the expected sojourn time of each queue to T_0, we seek to derive the necessary amount of upload bandwidth to serve each chunk.

The equilibrium of a Jackson network is characterized by the conditions in (1), when the individual queues achieve their

[1]Note that servers in a multiple-server queue in queueing theory is different from servers in a VoD application or a cloud.

[2]Note that the average Poisson arrival rate is fixed only at one time, to derive the server capacity demand in one time interval. In our practical cloud provisioning algorithm in Sec. V-B, the average arrival rate is dynamically learned and varying over time, in order to capture the burstiness of user arrivals at different times.

respective equilibrium.

$$\begin{cases} \lambda_1^{(c)} = \alpha \cdot \Lambda^{(c)} + \sum_{j=1}^{J^{(c)}} \lambda_j^{(c)} \cdot P_{j1}^{(c)}, \\ \lambda_i^{(c)} = \frac{1-\alpha}{J^{(c)}-1} \cdot \Lambda^{(c)} + \sum_{j=1}^{J^{(c)}} \lambda_j^{(c)} \cdot P_{ji}^{(c)}, i = 2, \ldots, J^{(c)}, \\ \rho_i^{(c)} = \frac{\lambda_i^{(c)}}{\mu} < m_i^{(c)}. \end{cases} \quad (1)$$

Here $\rho_i^{(c)} = \frac{\lambda_i^{(c)}}{\mu}$ is the server utilization in queue $Q_i^{(c)}$.

To derive the required service rates of the queues for smooth playback, we first derive the expected number of users in each queue in equilibrium, and then apply Little's law.

Let $p_i^{(c)}(k)$ denote the probability that k users are in chunk queue $Q_i^{(c)}$. Using the standard queuing analysis together with Erlang's C-Formula [23], we derive the equilibrium distribution of users in the queues as

$$\begin{aligned} p_i^{(c)}(0) &= \Big(\sum_{k=0}^{m_i^{(c)}-1} \frac{{\rho_i^{(c)}}^k}{k!} + \frac{m_i^{(c)} \cdot {\rho_i^{(c)}}^{m_i^{(c)}}}{m_i^{(c)}!(m_i^{(c)} - \rho_i^{(c)})} \Big)^{-1}, \\ p_i^{(c)}(k) &= \begin{cases} p_i^{(c)}(0) \cdot \frac{{\rho_i^{(c)}}^k}{k!}, & (0 \le k \le m_i^{(c)}) \\ p_i^{(c)}(0) \cdot \frac{{\rho_i^{(c)}}^k}{m_i^{(c)}! \cdot {m_i^{(c)}}^{k-m_i^{(c)}}} & (k > m_i^{(c)}) \end{cases}, \\ & i = 1, \ldots, J_i^{(c)}. \end{aligned} \quad (2)$$

Then, we can derive the expected number of users in queue $Q_i^{(c)}$ as

$$\begin{aligned} &\mathbb{E}(n_i^{(c)}) = \textstyle\sum_{k=0}^{\infty} k \cdot p_i^{(c)}(k) \\ &= p_i^{(c)}(0) \cdot [\textstyle\sum_{k=0}^{m_i^{(c)}} k \cdot \frac{{\rho_i^{(c)}}^k}{k!} \\ &\quad + \frac{{m_i^{(c)}}^{m_i^{(c)}}}{m_i^{(c)}!} \cdot W^{m_i^{(c)}+1} \cdot (m_i^{(c)} + 1 + \frac{W}{1-W})], \end{aligned} \quad (3)$$

where $W = \frac{\rho_i^{(c)}}{m_i^{(c)}}, i = 1, \ldots, J_i^{(c)}$.

If the average sojourn time in each queue $Q_i^{(c)}$ is T_0, we have $\mathbb{E}(n_i^{(c)}) = \lambda_i^{(c)} T_0$, according to Little's Law. Given $\lambda_i^{(c)}$ and T_0, based on this equation and Eqn. (3), we can derive the expected number of servers, $m_i^{(c)}$, in queue $Q_i^{(c)}$, which guarantees smooth playback. In particular, $m_i^{(c)}$'s are calculated as follows. We first derive $\lambda_i^{(c)}$'s using Eqn. (1) and then $\lambda_i^{(c)} T_0$ is known. We then derive $m_i^{(c)}$'s in an iterative fashion: we initialize $m_i^{(c)}$ to 1, and increase its value each step until $\mathbb{E}(n_i^{(c)})$ becomes equal to $\lambda_i^{(c)} T_0$.

The total upload bandwidth needed to serve chunk i is therefore $s_i^{(c)} = R m_i^{(c)}$. With the client-server implementation, this is the amount of upload capacity, $\Delta_i^{(c)}$, the cloud infrastructure needs to supply to serve chunk i, for $i = 1, \ldots, J_i^{(c)}$.

C. P2P VoD

In a P2P VoD application, the required upload bandwidth to serve each chunk i (*i.e.*, $s_i^{(c)} = R m_i^{(c)}$), comes from two sources: upload capacity $\Delta_i^{(c)}$ provisioned by the cloud, and upload bandwidth $\Gamma_i^{(c)}$ from peers in the channel who own chunk i. We have $R m_i^{(c)} = \Delta_i^{(c)} + \Gamma_i^{(c)}$. To derive the capacity needed from the cloud, we study the total upload bandwidth that can be provided from the peers, in a P2P VoD system based on a typical rarest-first scheduling scheme.

The P2P VoD scheme: A tracker server maintains lists of peers in each video channel and the chunks they buffer. A peer obtains the list of peers from the tracker who have the chunks it wishes to download, and requests the chunks from these peers. In response to the requests, a peer always serves chunks according to their rareness, *i.e.*, requests for the rarest chunk are served first, and then those for the less rare chunk, and so on, as many as its upload bandwidth can accommodate. The rareness of chunks is provided by the tracker, based on the number of peers currently owning each chunk.

Based on the above scheme, we first derive the expected number of peers who buffer chunk i ($i = 1, \ldots, J_i^{(c)}$) in the equilibrium state, and then study the bandwidth supplied by those peers for uploading the chunk.

Let $\nu_i^{(c)}$ denote the number of peers in channel c that have chunk i in their buffers. It is the sum of the numbers of peers who have previously downloaded the chunk and are currently in other chunk queues. The peers in queue $Q_i^{(c)}$ are still downloading chunk i, and we do not consider them as suppliers of the chunk.

Let $\nu_{ij}^{(c)} (j \neq i)$ denote the number of peers in chunk queue $Q_j^{(c)}$ that have buffered chunk i. We use $\nu_{ii}^{(c)}$ to denote the number of peers in queue $Q_i^{(c)}$ (who can supply chunk i upon departure from the queue), and thus $\mathbb{E}(\nu_{ii}^{(c)}) = \mathbb{E}(n_i^{(c)})$.

Proposition 1: The expected number of peers in chunk queue $Q_j^{(c)}$ in the equilibrium state, which have buffered chunk i, is

$$\mathbb{E}(\nu_{ij}^{(c)}) = \sum_{l=1}^{J^{(c)}} \mathbb{E}(\nu_{il}^{(c)}) \cdot P_{lj}^{(c)}, \quad \forall i = 1, \ldots, J^{(c)}, \forall j \neq i.$$

Due to space constraint, interested readers are referred to our technical report [24] for the detailed proof of the proposition.

Since $\mathbb{E}(n_i^{(c)})$ can be derived using Eqn. (3), we can derive $\mathbb{E}(\nu_{ii}^{(c)}) = \mathbb{E}(n_i^{(c)})$, based on which $\mathbb{E}(\nu_{ij}^{(c)}), \forall j \neq i$, can be calculated using Proposition 1. Then the expected total number of peers in channel c who have chunk i is derived as

$$\mathbb{E}(\nu_i^{(c)}) = \textstyle\sum_{j=1, j\neq i}^{J^{(c)}} \mathbb{E}(\nu_{ij}^{(c)}). \quad (4)$$

We next study the amount of upload bandwidth supplied by those peers with chunk i to serve the chunk, denoted as $\Gamma_i^{(c)}, i = 1, \ldots, J_i^{(c)}$. In the following analysis, each peer is assumed to have the same upload bandwidth of u (the analysis can be readily extended to cases with heterogeneous bandwidths).

We sort the chunks in increasing order of $\mathbb{E}(\nu_i^{(c)})$, *i.e.*, decreasing order of chunk rareness, and denote the ordered chunk sequence as $\{\pi_1, \ldots, \pi_{J^{(c)}}\}$, where π_1 represents the rarest chunk. Let $\Psi(\pi_j, \pi_k)$ denote the probability that a peer simultaneously owns chunks π_j and π_k. Based on a simplified assumption that each of the $\mathbb{E}(\nu_{\pi_k}^{(c)})$ peers that own chunk π_k supplies an equal share of the total upload bandwidth $\Gamma_{\pi_k}^{(c)}$ for the chunk, we derive the expected amount of peer upload bandwidth contribution as

$$\mathbb{E}(\Gamma_{\pi_k}^{(c)}) = \begin{cases} min\{m_{\pi_1}^{(c)} \times r, \mathbb{E}(\nu_{\pi_1}^{(c)}) \times u\}, & k = 1, \\ min\{m_{\pi_k}^{(c)} \times r, \mathbb{E}(\nu_{\pi_k}^{(c)}) \times u - \\ \quad \textstyle\sum_{j=1}^{k-1} [\Psi(\pi_j, \pi_k) \times \sum_{l=1}^{J^{(c)}} \mathbb{E}(n_l^{(c)}) \times \frac{\mathbb{E}(\Gamma_{\pi_j}^{(c)})}{\mathbb{E}(\nu_{\pi_j}^{(c)})}]\}, \\ k = 2, \ldots, J^{(c)}. \end{cases} \quad (5)$$

The rationale behind the above formula is as follows.

For the rarest chunk π_1, all peers with the chunk will maximally allocate bandwidth to serve it, and thus the overall

peer bandwidth contribution $\mathbb{E}(\Gamma_{\pi_1}^{(c)})$ is the minimal between the total upload bandwidth from those peers, $\mathbb{E}(\nu_{\pi_1}^{(c)}) \times u$, and the bandwidth demand to address its download requests, $m_{\pi_1}^{(c)} \times r$.

For another chunk π_k, the upload capacity that can be supplied from peers that own the chunk, equals the total upload bandwidth from those peers, $\mathbb{E}(\nu_{\pi_k}^{(c)}) \times u$, minus their bandwidth already allocated to other rarer chunks. The deduction part is calculated as follows: for each chunk π_j which is rarer than π_k ($j < k$), $\Psi(\pi_j, \pi_k) \times \sum_{l=1}^{J^{(c)}} \mathbb{E}(n_l^{(c)})$ is the number of peers in the channel who concurrently own both chunks, and the bandwidth each of them contributes to serve chunk π_j is $\frac{\mathbb{E}(\Gamma_{\pi_j}^{(c)})}{\mathbb{E}(\nu_{\pi_j}^{(c)})}$, based on the simplified assumption above.

The probability that a peer simultaneously owns two chunks π_j and π_k, *i.e.*, $\Psi(\pi_j, \pi_k)$, can be calculated by summing up the probabilities of all possible sequences of chunk queue transitions, which include queue $Q_{\pi_j}^{(c)}$ and queue $Q_{\pi_k}^{(c)}$. Due to space constraint, interested readers are referred to our technical report [24] for the detailed steps.

With the amount of peer bandwidth contribution computed using Eqn. (5), we can eventually derive the expected upload capacity that the cloud needs to supplement for uploading chunk i, as $\mathbb{E}(\Delta_i^{(c)}) = Rm_i^{(c)} - \mathbb{E}(\Gamma_i^{(c)})$, for $i = 1, \ldots, J_i^{(c)}$.

V. Cloud Provisioning Algorithm

We now design a dynamic provisioning algorithm which the VoD provider would execute when requesting the needed cloud resources. The algorithm makes use of the demand derived in the previous section. We first formulate two optimization problems to characterize optimal VM request and storage rental, and then design the cloud provisioning algorithm.

A. Optimal VM and Storage Rental

To deploy a VoD server application on the cloud infrastructure presented in Sec. III, the VoD provider needs to request a certain amount of storage to store its videos, as well as a number of VMs to serve the chunks from the storage. Based on the equilibrium demand $\mathbb{E}(\Delta_i^{(c)}), i = 1, \ldots, J^{(c)}, c = 1, \ldots, C$, derived in Sec. IV and the VoD provider's budget, the VM and storage configuration can be formulated into two optimization problems. Our discussion in this subsection apply to both client-server-based and P2P VoD applications.

1) Storage Rental: Let constant vector $\{u_1, \ldots, u_F\}$ represent the performance factors for NFS cluster $1, \ldots, F$, respectively, where a larger u_f for a cluster means a higher performance level (*e.g.* larger I/O throughput). Binary variable $x_{if}^{(c)}$ indicates that chunk i in channel c is to be deployed in NFS cluster f with $x_{if}^{(c)} = 1$, and not with $x_{if}^{(c)} = 0$. Let S_f be the available storage capacity of cluster f in bytes, and p_f be the storage cost per byte per unit time on f. $\mathbb{B}_S$ denotes the storage budget per unit time the VoD provider is willing to afford. Recall the size of each chunk is rT_0. The optimal storage rental problem, to decide which NFS cluster each chunk in each video should be deployed onto, is formulated as:

$$\max \sum_{c=1}^{C} \sum_{i=1}^{J^{(c)}} \sum_{f=1}^{F} u_f \Delta_i^{(c)} x_{if}^{(c)}$$
$$s.t. \begin{cases} \sum_{f=1}^{F} x_{if}^{(c)} = 1, \quad \forall i = 1, \ldots, J^{(c)}, c = 1, \ldots, C, \\ \sum_{c=1}^{C} \sum_{i=1}^{J^{(c)}} x_{if}^{(c)} \le \frac{S_f}{rT_0}, \quad \forall f = 1, 2, \ldots, F, \\ \sum_{c=1}^{C} \sum_{i=1}^{J^{(c)}} \sum_{f=1}^{F} p_f r T_0 x_{if}^{(c)} \le \mathbb{B}_S, \\ x_{if}^{(c)} = \{0, 1\}, \forall i = 1, \ldots, J^{(c)}, \\ \qquad\qquad \forall c = 1, \ldots, C, f = 1, \ldots, F. \end{cases} \tag{6}$$

The objective function maximizes the aggregate performance for retrieving all chunks in all videos from the NFS storage system, where $\Delta_i^{(c)} x_{if}^{(c)}$ represents the aggregate demand of chunk i in channel c from cluster f. With the first constraint, we restrict that only one copy of each chunk is to be deployed in the storage system, since all VMs can access all NFS servers. The second and third constraints represent the storage capacity and budget constraints, respectively. The optimization problem in (6) is a Knapsack-like problem [25]. We design an efficient heuristic to derive the approximation solution:

Storage rental heuristic: Sort all chunks in all channels in decreasing order of $\Delta_i^{(c)}, i = 1, \ldots, J^{(c)}, c = 1, \ldots, C$, and sort the NFS clusters in decreasing order of the marginal utility per unit cost $\frac{u_f}{p_f}, f = 1, \ldots, F$. Starting with the chunk with the highest demand, we store it in the best NFS cluster (with the largest $\frac{u_f}{p_f}$) as long as the cluster is not full, or move on to the second best cluster otherwise. This process repeats for all the chunks in the ordered list, as long as the total storage budget spent does not exceed $\mathbb{B}_S$.

With this heuristic, we seek to place the most popular chunks on the NFS clusters at the highest performance level with the most economic budget expenditure. We note that if the budget runs out when not all the chunks have been stored, the optimization problem does not have a feasible solution, which signals to the VoD provider that their set budget is not feasible given the current storage prices, which should be increased.

2) VM Configuration: Let constant vector $\{\tilde{u}_1, \ldots, \tilde{u}_V\}$ represent the performance factors for VMs in virtual cluster $1, \ldots, V$, respectively, where a larger $\tilde{u}_v$ for a cluster means a higher-grade configuration (*e.g.*, CPU, I/O). We define variable $z_{iv}^{(c)}$ as the number of VMs to request from virtual cluster v, to serve chunk i in channel c. Let N_v be the maximal number of available VMs cluster v can provision, and $\tilde{p}_v$ be the rental cost per unit time of one VM from cluster v. $\mathbb{B}_M$ denotes the VM rental budget per unit time from the VoD provider. Recall R is the bandwidth each VM is allocated. The optimal VM configuration problem, to decide how many VMs per virtual cluster the VoD provider should request, is formulated as:

$$\max \sum_{c=1}^{C} \sum_{i=1}^{J^{(c)}} \sum_{v=1}^{V} \tilde{u}_v z_{iv}^{(c)}$$
$$s.t. \begin{cases} \sum_{v=1}^{V} z_{iv}^{(c)} = \frac{\Delta_i^{(c)}}{R}, \quad \forall i = 1, 2, \ldots, J^{(c)}, c = 1, \ldots, C, \\ \sum_{c=1}^{C} \sum_{i=1}^{J^{(c)}} z_{iv}^{(c)} \le N_v, \quad \forall v = 1, \ldots, V, \\ \sum_{c=1}^{C} \sum_{i=1}^{J^{(c)}} \sum_{v=1}^{V} \tilde{p}_v z_{iv}^{(c)} \le \mathbb{B}_M. \end{cases} \tag{7}$$

The objective function maximizes the aggregate performance for serving all chunks of all videos from the VMs

requested. The first constraint states that the total upload bandwidth of VMs requested for each chunk should be sufficient to serve the demand for the chunk. The second and third constraints represent the VM number and budget constraints. We design the following heuristic to solve the optimization problem in (7):

VM configuration heuristic: Sort VM clusters in decreasing order of the marginal utility per unit cot $\frac{\tilde{u}_v}{\tilde{p}_v}, v = 1, \ldots, V$. For any chunk i in channel c, the total number of VMs it needs is $\frac{\Delta_i^{(c)}}{R}$, and we allocate as many VMs as possible to serve this demand from the best virtual cluster (with the largest $\frac{\tilde{u}_v}{\tilde{p}_v}$) if it still has available VMs, or move on to the second best VM cluster otherwise. This process repeats for all chunks, as long as the total VM rental budget $\mathbb{B}_M$ is not exceeded.

With this heuristic, we seek to maximally place chunks on the virtual clusters with the best configurations at the modest budget. Note that $z_{iv}^{(c)}$ can be fractional: its integer part corresponds to the number of VMs which will be entirely used to serve chunk i, and the fractional part indicates the fraction of bandwidth used to serve chunk i at a shared VM, which may concurrently serve multiple chunks. If one VM is used to serve more than one chunk, we will maximally allow consecutive chunks in one channel to be served by the VM. Similarly, if the VM rental budget is exceeded when not all the chunk demand has been served, the optimization problem is not feasible, and the VoD provider should increase the budget accordingly.

B. Dynamic Cloud Provisioning Algorithm

We now propose a practical algorithm through which the cloud and VoD providers would cooperate to implement our VoD-on-cloud system, *CloudMedia*. We illustrate our algorithm with the case of a P2P VoD application, and the algorithm can be easily adapted to client-server VoD applications.

To start, the VoD provider deploys its videos to the NFS cluster and the server application to VMs in the cloud infrastructure, where the amount of storage and the number of VMs are estimated using the storage and VM rental heuristics presented in Sec. V-A, and based on the application's empirical user scale and viewing pattern information and the equilibrium demand derived. Overtime, the VoD provider dynamically adjusts its cloud resource requests based on the current demand. As hourly resource rental is commonly supported in state-of-the-art cloud systems [12], we assume our provisioning algorithm below is periodically run every interval of $T = 1$ hour. Key modules in the algorithm are illustrated in Fig. 3.

The **tracking server** maintains peer lists for each video and the chunks they are caching, as well as the IP addresses and ports of the entry points to the cloud infrastructure (*i.e.*, public access addresses of the cloud). When a peer first joins or seeks to a new playback position in a channel, it asks for neighbors from the tracking server which returns a list of peers who have the required chunks. If there is insufficient peer supply, the tracking server will return a 3-tuple, *i.e.*, <IP address of a cloud entry point, a list of port numbers, a ticket> to the peer. Then the peer can send its chunk requests to the cloud. Once the ticket is verified at the entry point, the requests will be forwarded to the VMs in the cloud which will then serve the requested chunks using the port-forwarding technique. A VM will send a required chunk directly to the peer.

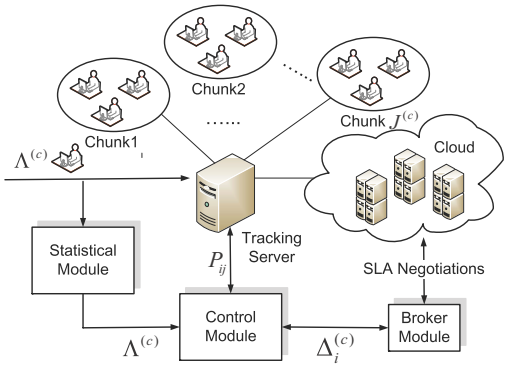

Fig. 3. *CloudMedia*: an illustration of the key modules.

During each interval T, the tracking server summarizes the average user arrival rate $\Lambda^{(c)}$ to each channel $c = 1, \ldots, C$, as well as the viewing patterns $P_{ij}^{(c)}$ for each channel. It then sends these statistics to the controller at the end of the interval.

Using the collected arrival rates and viewing patterns, the **controller** estimates the equilibrium demand for upload capacity to serve each chunk, *i.e.*, $s_i^{(c)} = Rm_i^{(c)}, i = 1, \ldots, J^{(c)}, c = 1, \ldots, C$, using the analytical method in Sec. IV-B, and the expected amount of peer upload bandwidth contribution for each chunk, $\mathbb{E}(\Gamma_i^{(c)})$, based on the method in Sec. IV-C. The expected amount of upload capacity to be provisioned from the cloud is therefore $\mathbb{E}(\Delta_i^{(c)}) = Rm_i^{(c)} - \mathbb{E}(\Gamma_i^{(c)})$.

The controller then negotiates with the cloud provider via the **broker**, for prices of VM and storage rental and QoS of the resources. When the SLA is set and information on the virtual and NFS clusters is provided (*e.g.*, prices, current availability), the controller computes in details its VM requests for each chunk, applying the heuristics in Sec. V-A2, according to its VM rental budget. If there are new videos to deploy or if the demand for chunks has changed significantly since last interval, the controller may also recompute the NFS storage rental using the heuristic in Sec. V-A1. Then, it sends the change requests to the cloud via the broker.

After the requests are received by the **request monitor** in the cloud infrastructure (shown in Fig.1), the **VM scheduler** and the **NFS scheduler** adjust their VM and NFS server provisioning accordingly.

In our dynamic provisioning algorithm, user arrival patterns in the previous time interval (hour) are used to predict the capacity demand in the next interval. This design achieves implementation simplicity, and has been validated by our evaluation results, the cloud resources provisioned based on the predicted equilibrium demand serve the actual demand quite well. Nevertheless, more accurate prediction method based on historical data collected over more intervals can be applied for better performance, which however is not the focus of the current paper and can be treated in our future work.

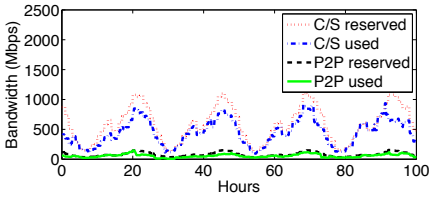

Fig. 4. Cloud capacity provisioning vs. usage.

VI. Performance Evaluation

We verify our analytical results and evaluate the performance of the dynamic cloud provisioning algorithm, based on large-scale experiments using a cloud system and a prototype VoD application we have implemented and deployed on a cluster of machines.

A. Prototype Implementation and Experimental Settings

We have built a cloud infrastructure and a VoD system using 100+ commodity computers (Intel(R) Pentium(R) 4 CPU 2.80GHz, 1G RAM, and 80G hard drive), obsoleted from student laboratories in the Computer Science Department at the University of Hong Kong. The computers are interconnected via a collection of IBM 8275 Ethernet switches (Model 324). We divide the computers into 3 groups, one for constructing the cloud infrastructure (about 50 computers), one for emulating the VoD user network (about 50 computers), and the other with 3 computers for implementing control mechanisms (*e.g.*, tracking server and the controller module).

The computers allocated for cloud services are further divided into virtual clusters and NFS clusters. On each physical machine in a virtual cluster, we install Xen hypervisor VMs [26] with Fedora 8 as the hosting operating system. On each Xen VM, we install CentOS 5.4 as the guest operating system and a modified Apache server (based on version 2.2.17) to provide the streaming server service. A light-weighted cloud management system to achieve the functional modules in Sec. III-A is implemented using Java, which features a user-friendly management GUI, rapid launch, allocation, and shutdown of VMs based on user demands, real-time performance monitoring and load balancing among VMs, etc.

On each of the computers allocated for VoD network, we run tens of concurrent VoD clients (users). Each VoD client is implemented in Java and executed as one process. In the client-server mode of the VoD application, the users directly connect to server services in the cloud; in the P2P mode, the peers in each channel interconnect into a mesh overlay and resort to the cloud only when necessary. Media chunks are delivered over TCP connections among the cloud and the users.

We create 3 VM clusters and 2 NFS clusters with different configurations given in Table II and Table III, respectively. Each VM in the VM clusters is allocated a fixed bandwidth of 10Mbps. The prices are set based on the charging model of Amazon EC2 [12] and S3 [13]. The VM and storage rental budgets are $\mathbb{B}_M = \$100$ per hour and $\mathbb{B}_S = \$1$ per hour, respectively.

We deploy 20 video channels with different popularities following a Zipf-like distribution with the total number of concurrent online peers around 2500. The streaming playback rate of each channel is $r = 50$ Kbytes/s (400 Kbps) and the length of each video is 100 minutes. The size of each chunk is 15 Mbytes, corresponding to a playback time of $T_0 = 5$ minutes.[3] To emulate realistic VoD user dynamics, we have generated a synthetic trace, following the measured user dynamics and other characteristics in PPLive VoD as discussed in [21]. Specifically, user population in each channel follows a daily pattern with two flash crowds around noon and in the evening, respectively. The interval between two playback jumps made by a VoD user follows an exponential distribution with an expected length of 15 minutes. The upload capacity of users follows a Pareto distribution within range [180Kbps, 10Mbps] with shape parameter $k = 3$, which is implemented via bandwidth control in VoD client processes.

B. Streaming Performance

We emulate the execution of user swarms together with *CloudMedia* system over one week's time, and plot in Fig. 4 the provisioned upload capacity from the cloud infrastructure and the actually used cloud upload capacity by the users, in both the cases of client-server and P2P VoD implementations. We observe that in the majority of time, provisioned bandwidth is larger than the used, showing the effectiveness of our server demand prediction: even if simplified assumptions have been made in our modeling, experimental results under realistic settings have exhibited good matching between user demand and cloud supply, even at times of flash crowds. In addition, the amount of cloud capacity needed in P2P VoD is much smaller than that in client-server VoD, showing that peer-assisted implementation can further significantly alleviate the operational cost of VoD providers, who have already exploited the cost-efficient cloud paradigm.

Fig. 5 shows the average streaming quality in the 20 streaming channels, computed as the percentage of users in all the channels with smooth playback in the past 5 minutes. The streaming quality in P2P VoD is slightly worse than that in client-server VoD (but still achieves an average of 0.95), which represents a minor tradeoff between streaming quality and server (cloud) cost with a peer-assisted implementation.

We next take a closer look at the streaming performance in each of the streaming channels. In Fig. 6, we plot the streaming quality vs. the number of users in each channel in client-server VoD. The samples plotted are sizes of the 20 channels during one day's period of time (note that the size of each channel varies over time). We see that the streaming quality is generally good regardless of channel sizes. The results for P2P VoD are slightly worse, which we omit from the figure, as they significantly overlap with the results of client-server VoD.

[3]The selection of chunk size should aim to minimize the unnecessary number of times of VM switching during users' playback, while considering the average length of continuous playback between two VCR operations as well as the actual transmission efficiency. We have experimented with different chunk sizes and identified the one presented here as the best.

TABLE II
VIRTUAL CLUSTER CONFIGURATIONS

Type	Utility ($\tilde{u}_v$)	Memory	CPU	Hard Disk	Price ($\tilde{p}_v$) per hour	No. of VMs per cluster (N_v)
Standard	0.6	128 MB	500MHz	5GB	$0.450	75
Medium	0.8	192 MB	500MHz	5GB	$0.700	30
Advanced	1	256 MB	500MHz	5GB	$0.800	45

TABLE III
NFS CLUSTER CONFIGURATIONS

Type	Utility(u_f)	Rotation Speed	Price(p_f) (per GB per hour)	Capacity(S_f)
Standard	0.8	7200 RPM	$\$1.11 \times 10^{-4}$	20 GB
High	1	10800 RPM	$\$2.08 \times 10^{-4}$	20 GB

In the companion figure of Fig. 7, we plot the bandwidth provisioned to each channel (from the cloud) against the current size of the channel. We observe that bandwidth demand linearly increases with the number of users in a channel in client-server VoD, but scales very well with P2P VoD.

C. VM and Storage Usage

We next investigate the efficiency of VM startup and shutdown, the effectiveness of our storage and VM configuration heuristics, as well as the costs involved in operating the cloud.

In our implementation, VM instances are pre-deployed (and in "off" state) in the physical machines in the cloud infrastructure, corresponding to 3 different configurations given in Table II. When the *CloudMedia* system is in operation, VMs are launched and shut down by the cloud management system according to real-time VoD users' demand, using the dynamic provisioning algorithm in Sec. V. It takes around 25 seconds to turn on a VM, and even less time to shut it down. As VMs can be launched (or shut down) in parallel, latency involved in VM provisioning is small (at seconds), which enables timely service provisioning for a VoD application.

To evaluate the effectiveness of our storage rental and VM configuration heuristics, we select 4 channels with different average user numbers of $10, 60, 200, 600$, respectively, and compute the aggregate storage utility ($\sum_{i=1}^{J^{(c)}} \sum_{f=1}^{F} u_f \Delta_i^{(c)} x_{if}^{(c)}$) and aggregate VM utility ($\sum_{i=1}^{J^{(c)}} \sum_{v=1}^{V} \tilde{u}_v z_{iv}^{(c)}$) in each channel at different times, in the case of P2P VoD. In our experiments, the performance factors $\tilde{u}_v$ and u_f reflect the different memory allocation and hard disk speeds of different VM and storage clusters, respectively. Therefore, the aggregate utilities represent overall I/O performance at the allocated VMs and storage servers, respectively. The evolution of the utility values in Fig. 8 and 9 shows the adaptiveness of our heuristics, which always strives to achieve the best storage and VM allocation for chunks (and channels) according to their current popularity.

Fig. 10 gives the total VM rental cost to support the VoD system in one day's period of time, in the cases of P2P VoD and client-server VoD, respectively. We observe that the average cost of VM rental with P2P VoD is about $\$4.27$ per hour, and that for client-server VoD is much larger at an average of $\$48$ per hour, which also varies significantly over time due to the dynamics of user population. This illustrates the great potential of using a hybrid P2P and cloud paradigm in providing high-performance streaming with low cost.

On the other hand, the storage cost for NFS rental can almost be ignored, at around $\$0.018$ per day. Since our costs are derived based on practical pricing models from [12] [13], this verifies that the cost of deploying a large-scale VoD application on a cloud infrastructure largely lies at the VM rentals, instead of storing the many videos onto the cloud storage.

D. Impact of Peer Bandwidth Sufficiency

We have also evaluated the impact of peers' upload bandwidth availability on cloud capacity provisioning and streaming quality, in the case of P2P VoD implementation. With different experiments, we vary the ratio of average upload capacity per peer over the streaming rate r. As expected, less cloud resource is needed when peer average upload capacity is larger, whose plots we omit as the results are quite intuitive.

We plot in Fig. 11 the evolution of average streaming quality in the system at different peer average bandwidth levels. The streaming qualities are satisfactory in all cases, showing that our cloud capacity provisioning can well absorb different bandwidth demand from the P2P overlay over time, no matter whether the peer bandwidth contribution is sufficient or not.

VII. CONCLUDING REMARKS

This paper introduces the paradigm of utilizing cloud services to support large-scale Internet-based applications. Using the example of video-on-demand applications, we demonstrate how on-demand cloud resource provisioning can desirably meet the dynamic and intensive resource demands of VoD over the Internet. Our main contributions are: First, we propose a novel queueing network model to characterize users' viewing behaviors, with which we derive the equilibrium demand of upload bandwidth for smooth playback for both client-server and P2P VoD implementations. Second, taking practical cloud parameters into account, we formulate two optimization problems related to VM provisioning and storage rental, for which we propose some efficient solutions. Third, a practical dynamic cloud provisioning algorithm is designed and implemented, by which a VoD provider can effectively configure the cloud services to meet its demands.

Our extensive performance evaluations based on real system implementations adopt practical user dynamics observed in real-world VoD systems, and the results confirmed the adaptability and effectiveness of *CloudMedia* in handling time-

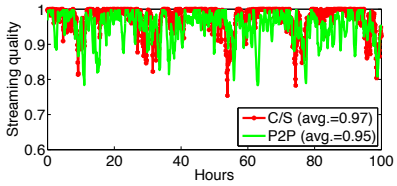

Fig. 5. Average streaming quality in the VoD system.

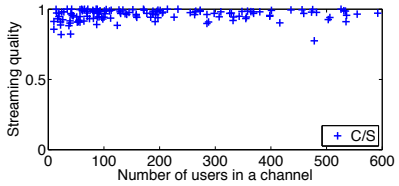

Fig. 6. Channel streaming quality vs. channel size for all channels in one day's time.

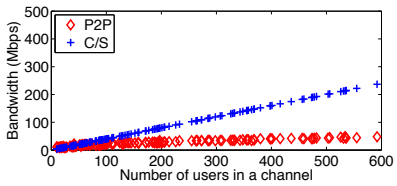

Fig. 7. Cloud capacity provisioning vs. channel size for all channels in one day's time.

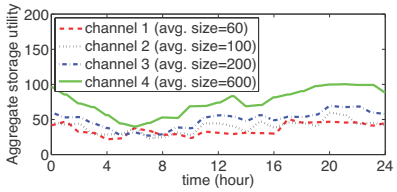

Fig. 8. Evolution of aggregate storage utility in 4 representative channels.

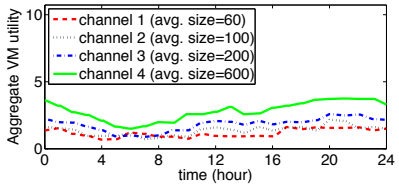

Fig. 9. Evolution of aggregate VM utility in 4 representative channels.

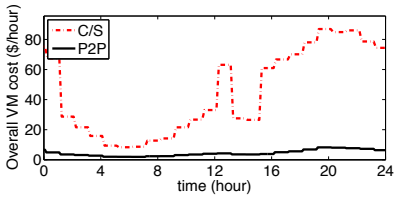

Fig. 10. Evolution of overall VM rental cost.

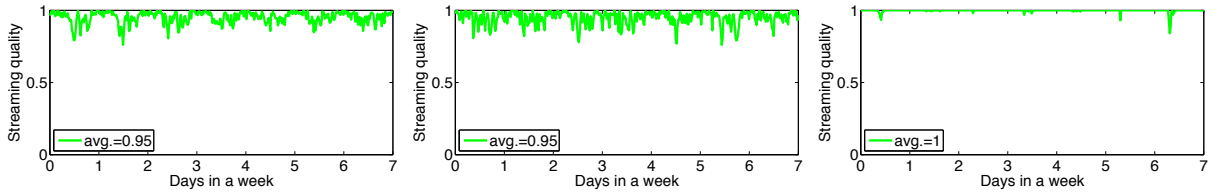

Fig. 11. Average streaming quality with P2P VoD implementation, at different ratios of peer average upload capacity over the streaming rate: (1) 0.9, (2) 1, (3) 1.2.

varying demands and guaranteeing smooth playback at any time. It can be observed that the combination of cloud and the P2P paradigm can achieve ultimate scalability for Internet-based applications with minimum operational costs. In our ongoing work, we are expanding to cloud systems spanning different geographic locations, as well as more extensive evaluations with Internet-wide deployment.

REFERENCES

[1] *Cloud Computing*, http://csrc.nist.gov/groups/SNS/cloud-computing.

[2] M. Armbrust, A. Fox, R. Grifth, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. P. A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," *Technical report*, 2009.

[3] S. Pandey, L. Wu, S. Guru, and R. Buyya, "A Particle Swarm Optimization (PSO)-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environment," in *Proc. of IEEE AINA*, 2010.

[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems, Elsevier Science*, vol. 25, no. 6, pp. 599–616, June 2006.

[5] B. Cheng, X. Liu, Z. Zhang, H. Jin, L. Stein, and X. Liao, "Evaluation and Optimization of a Peer-to-Peer Video-on-Demand System," *J. Syst. Archit.*, vol. 54, no. 7, pp. 651–663, Jul. 2008.

[6] Z. Liu, C. Wu, B. Li, and S. Zhao, "UUSee: Large-Scale Operational On-Demand Streaming with Random Network Coding," in *Proc. of IEEE INFOCOM*, March 2010.

[7] Y. Xiao, C. Lin, Y. Jiang, X. Chu, and S. Shen, "Reputation-based QoS Provisioning in Cloud Computing via Dirichlet Multinomial Model," in *Proc. of IEEE ICC*, 2010.

[8] Peixoto, M. Santana, M. Estrella, J. Tavares, T. Kuehne, B. Santana, and R.H.C., "A Metascheduler Architecture to Provide QoS on the Cloud Computing," in *Proc. of IEEE ICT*, 2010.

[9] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing," in *Proc. of IEEE INFOCOM*, 2010.

[10] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," in *Proc. of IEEE INFOCOM*, 2010.

[11] R. Urgaonkar, U. C. Kozat, K. Igarashi, and M. J. Neely, "Dynamic Resource Allocation and Power Management in Virtualized Data Centers," in *Proc. of IEEE/IFIP NOMS*, 2010.

[12] *Amazon Elastic Compute Cloud*, http://aws.amazon.com/ec2/.

[13] *Amazon Simple Storage Service*, http://aws.amazon.com/s3/.

[14] S. Liu, M. Chen, S. Sengupta, M. Chiang, J. Li, and P. A. Chou, "P2P Streaming Capacity under Node Degree Bound," in *Proc. of IEEE INFOCOM*, March 2010.

[15] D. Wu, Y. Liu, and K. W.Ross, "Queuing Network Models for Multi-Channel P2P Live Streaming Systems," in *Proc. of IEEE INFOCOM*, 2009.

[16] R. Kumar, Y. Liu, and K. Ross, "Stochastic Fluid Theory for P2P Streaming Systems," in *Proc. of IEEE INFOCOM*, 2007.

[17] N. Parvez, C. Williamson, A. Mahanti, and N. Carlsson, "Analysis of BitTorrent-like Protocols for On-Demand Stored Media Streaming," in *Proc. of ACM SIGMETRICS*, June 2008.

[18] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance Bounds for Peer-Assisted Live Streaming," in *Proc. of ACM SIGMETRICS*, June 2008.

[19] C. Wu, B. Li, and S. Zhao, "Multi-channel Live P2P Streaming: Refocusing on Servers," in *Proc. of IEEE INFOCOM*, 2008.

[20] F. Liu, Y. Sun, B. Li, and B. Li, "Quota: Rationing Server Resources in Peer-Assisted Online Hosting Systems," in *Proc. of IEEE ICNP*, 2009.

[21] Y. Huang, T. Z. J. Fu, D.-M. Chiu, J. C. S. Lui, and C. Huang, "Challenges, Design and Analysis of a Large-Scale P2P-VoD System," in *Proc. of ACM SIGCOMM*, August 2008.

[22] J. R. Jackson, "Jobshop-like Queueing Systems," *Management Science*, vol. 10, no. 1, pp. 131–142, 1963.

[23] Robert B. Cooper, *Introduction to Queueing Theory (2nd Edition)*. Elsevier North Holland, 1981.

[24] Y. Wu, C. Wu, B. Li, X. Qiu, and F. C. Lau, "CloudMedia: When Cloud on Demand Meets Video on Demand," http://i.cs.hku.hk/~ywu/papers/cloudmedia.pdf, CS, The University of Hong Kong, Tech. Rep., February 2011.

[25] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004.

[26] *Xen*, http://www.xen.org/.