Cloud Computing for Multimedia Services

19

The emergence of *cloud computing* [1] has dramatically changed the service models for modern computer applications. Utilizing elastic resources in powerful data centers, it enables end users to conveniently access computing infrastructure, platforms, and software provided by remote cloud providers (e.g., Amazon, Google, and Microsoft) in a pay-as-you-go manner or with long-term lease contracts. This new generation of computing paradigm, offering reliable, elastic, and cost-effective resource provisioning, can significantly mitigate the overhead for enterprises to construct and maintain their own computing, storage, and network infrastructures. It has provided countless new opportunities for both new and existing applications.

Existing applications, from content sharing and file synchronization to media streaming, have experienced a leap forward in terms of system efficiency and usability through leveraging cloud computing platforms. These advances mainly come from exploiting the cloud's massive resources with elastic provisioning and pricing and with smart computational offloading.

On the other hand, start-up companies can easily implement their novel ideas into real products with minimum investment in the initial stage and expand the system scale without much effort later on. A representative is Dropbox, a typical cloud storage and file synchronization service provider, which, from the very beginning, has relied on Amazon's S3 cloud servers for file storage and uses Amazon's EC2 cloud instances to provide such key functions as synchronization and collaboration among different users. We have also seen such new generation of multimedia services as cloud-based VoD and gaming that have emerged in the market and may change the whole business model in the coming years. A prominent example is Netflix, a major Internet streaming video provider, which migrated its infrastructure to Amazon's cloud platform in 2010 and has since become one of the most important cloud users. In total, Netflix has over 1 petabyte of media data stored in Amazon's cloud. It pays by bytes for bandwidth and storage resources, so that the long-term costs become much lower than those with over-provisioning in self-owned servers. Another example is Cloudcoder, an adaptive video transcoding service that offloads much of the processing to the cloud. Built on Microsoft's Azure cloud platform, Cloudcoder

Z.-N. Li et al., *Fundamentals of Multimedia*,
Texts in Computer Science, DOI: 10.1007/978-3-319-05290-8_19,

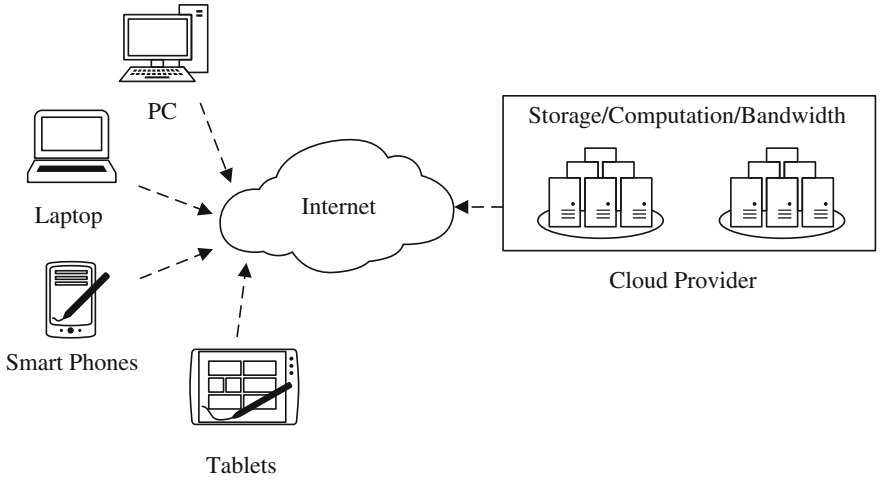

Fig. 19.1 A conceptual overview of cloud computing

can support a large number of transcoder instances simultaneously, which can also be automatically scaled to handle bursts of requests.

In this chapter, we provide an overview of cloud computing, focusing on its impact on multimedia services. We then discuss multimedia content sharing with cloud storage and multimedia computation offloading to the cloud. We also use cloud gaming as a case study to examine the role of the cloud in the new generation of interactive multimedia services.

19.1 Cloud Computing Overview

Cloud computing relies on sharing of resources to achieve coherence and economies of scale similar to a utility over a network, like the electricity grid does. As illustrated in Fig. 19.1, cloud users can run their applications on powerful server clusters offered by the cloud service provider, with system and development software readily deployed inside the cloud, mitigating the users' burden of full installation and continual upgrade on their local hardware/software. A cloud user can also store their data in the cloud instead of on their own devices, making ubiquitous data access possible.

At the foundation of cloud computing is the broader concept of resource virtualization and sharing. Focusing on maximizing the utilization of aggregated physical resources, cloud resources are shared by multiple users with dynamical on-demand allocation. For example, a cloud can serve European users during their business hours, while the same set of resources can later be used by North American users during their business hours. And each user sees its own dedicated virtual space.

The cloud services can be *public* or *private*. In a public cloud, the services and infrastructure are provided off-site over the public Internet. These clouds offer the greatest level of efficiency in resource sharing; however, they can be less secure and more vulnerable than private clouds. Unlike public clouds, in a private cloud, the services and infrastructure are maintained on a private network. These clouds offer the greatest level of security and control, though they require the company to still purchase and maintain the software and infrastructure. In either case, there are a common set of essential characteristics of cloud computing, as identified by the National Institute of Standards and Technology (NIST) [2]:

On-demand self-service. A user can unilaterally provision computing capabilities (e.g., server time and network storage) as needed without human interaction with each service provider;

Resource pooling and rapid elasticity. The provider's resources are pooled to serve multiple users, with different physical and virtual resources dynamically assigned and reassigned according to user demand. To a cloud user, the resources available for provisioning often appear unlimited and can be appropriated in any quantity at any time;

Measured service. Cloud systems automatically control and optimize resource use by leveraging a metering capability at an abstraction level appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). The resource usage can be monitored, controlled, and reported, providing transparency for both the provider and the users;

Broad network access. Persistent and quality network accesses are available to accommodate heterogeneous client platforms (e.g., mobile phones, tablets, laptops, and workstations).

In marketing, cloud services are mostly offered from data centers with powerful server clusters in three fundamental models (see Fig. 19.2): *Infrastructure as a service* (IaaS), *Platform as a service* (PaaS), and *Software as a service* (SaaS), where IaaS is the most basic and each higher model abstracts from the details of the lower models. Network as a Service (NaaS) and Communication as a Service (CaaS) have been recently added as part of the basic cloud computing models too, enabling a telecommunication-centric cloud ecosystem.

Infrastructure as a Service (IaaS)

IaaS is the very basic and essential cloud service. Well-known examples of IaaS include such infrastructure vendor environments as the Amazon's Elastic Compute Cloud (EC2), which allow users to rent virtual machines on which to run applications, and such cloud storage services as Amazon's Simple Storage Service (S3), which allow users to store and retrieve data, at any time, from anywhere on the Web.

In general, an IaaS provider offers a pool of computation resources, in the form of physical machines, or more often, *virtual machines*, as well as other resources, e.g., virtual-machine disk image library, data block or file-based storage, firewalls, load balancers, IP addresses, virtual local area networks, and software bundles. For wide-area connectivity, the users can use either the public Internet or dedicated virtual private networks.

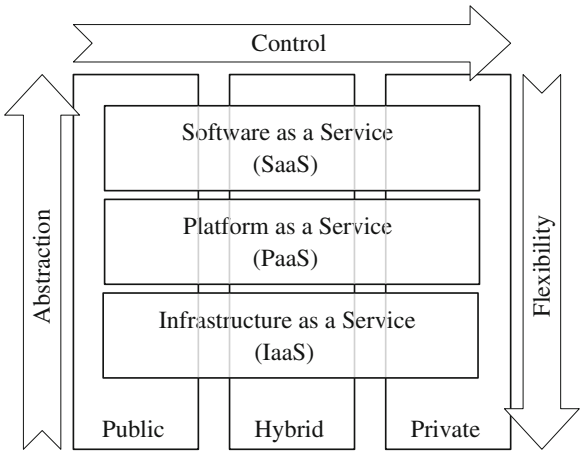

Fig. 19.2 An illustration of cloud service models

To deploy their applications, cloud users install operating-system images and their application software on the cloud infrastructure. With machine virtualization, an IaaS cloud provider can support a large numbers of users with its pool of hardware and software, and scale services up and down according to users' varying requirements. The cloud providers typically bill IaaS services on a utility computing basis, where the cost reflects the amount of resources allocated and consumed.

Platform as a Service (PaaS)
PaaS delivers development environments as a service, which typically includes the operating system, programming language execution environment, database, web server, etc. Applications can be built and run on the PaaS provider's infrastructure and then delivered to end users via the Internet. As such, the cost and complexity of purchasing and managing the underlying hardware and software layers can be greatly reduced. Moreover, the underlying computation and storage resources can scale automatically to match the applications' demands. Google's App Engine is a typical example of PaaS.

Software as a Service (SaaS)
SaaS, probably the most widely used cloud service model to date, allows an application to run on the infrastructure and platforms offered by the cloud rather than on local hardware/software. As such, the user of an application does not have to heavily invest on it own servers, software, license, etc. SaaS is usually priced on a usage basis, or with a monthly or yearly flat fee per user. The price is scalable and adjustable if users are added or removed at any point.

SaaS greatly reduces IT operational costs by outsourcing hardware and software maintenance and support to the cloud provider. This enables the business to reallocate IT operations costs away from hardware/software spending and personnel expenses, toward meeting other goals. Besides cost saving and simplified maintenance and support on the user's side, cloud applications also enjoy superior scalability, which can be achieved by cloning tasks onto multiple virtual machines at run-time

to meet changing work demands. A load balancer can distribute the workload over the virtual machines. Yet these are transparent to the cloud users, who see only the virtual machine allocated to itself. Google Apps and Microsoft Office 365 are typical examples of SaaS.

Figure 19.2 illustrates the relations among different cloud service models, particularly in terms of their abstraction, control, and flexibility levels.

Since year 2000, Amazon has played a leading role in the development of cloud computing by modernizing their data centers. In 2006, Amazon initiated a new product development effort to provide cloud computing to external customers, and launched the *Amazon Web Services* (AWS) on a utility computing basis, which has since become one of the most widely used cloud computing platforms. We next examine two representative services provided by Amazon's AWS, namely, S3 for storage and EC2 for computation, both of which have been widely used for supporting multimedia services.

19.1.1 Representative Storage Service: Amazon S3

Cloud storage has the advantage of being *always-on*, so that users can access their files from any device and can share their files with others who may access the content at an arbitrary time. With advanced storage management, cloud storage also provides a much higher level of reliability than local storage, yet with comparable or lower costs. All these are critical to media sharing, one of the most demanding storage services.

Amazon's S3 provides a web service interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the Web. It gives a developer access to the same highly scalable, reliable, secure, fast, inexpensive infrastructure that Amazon uses to run its own global network of websites. The service aims to maximize the benefits of scale and to pass those benefits on to developers and end users. To this end, S3 is intentionally built with a minimal feature set with simple operations.

An S3 user can write, read, and delete objects each containing from 1 byte to 5 terabytes of data. Each object is stored in a *bucket* and retrieved via a unique, developer-assigned key. The bucket can be stored in one of the several *regions*. The S3 user can choose a region to optimize for latency, minimize costs, or address regulatory requirements. S3 is currently available in the US Standard, US West (Oregon), US West (Northern California), EU (Ireland), Asia Pacific (Singapore), Asia Pacific (Tokyo), Asia Pacific (Sydney), South America (Sao Paulo), and GovCloud (US) regions. The objects stored in a region never leaves it unless transferred out, and, like a CDN routing, a network map is used to route the request. Figure 19.3 shows an example of a data object stored in the US West (Oregon) region.

S3 is built to be flexible so that protocols or functional layers can be easily added. The default download protocol is HTTP. A BitTorrent protocol interface is also provided to lower the costs for large-scale distribution.

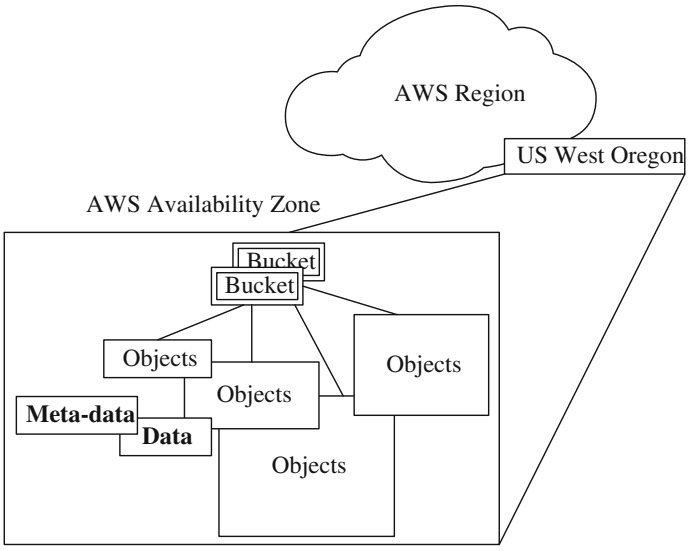

Fig. 19.3 An example of a data object stored in an Amazon AWS region (US East)

The data stored in Amazon S3 is secure by default; only the bucket and object owners have access to the S3 resources created by them. It supports multiple access control mechanisms, as well as encryption for both secure transit and secure storage on disk. To increase durability, it synchronously stores the data across multiple facilities, and calculates checksums on all network traffic to detect corruption of data packets when storing or retrieving data. Unlike traditional storage systems that require laborious data verification and manual repair, S3 performs regular, systematic data integrity checks and is built to be automatically self-healing.

S3 also automatically archives objects to even lower cost storage options or perform recurring deletions, reducing the costs over an object's lifetime. The costs can be monitored and controlled by the users through the S3 APIs or Management Console. Table 19.1 shows the current pricing plan of the S3 services.

19.1.2 Representative Computation Service: Amazon EC2

Amazon's *Elastic Compute Cloud* (EC2) is a web service that provides resizable compute capacity in the cloud. It presents a virtual computing environment, allowing users to launch instances with a variety of operating systems, load them with customized application environment, and manage network access permissions. Different instance provisioning plans are available to meet a user' demands:

On-Demand Instances: On-Demand Instances let the user pay for compute capacity by hour with no long-term commitments. This frees the user from the costs and complexities of planning, purchasing, and maintaining hardware/software and

Table 19.1 The storage pricing of US Standard Region

	Standard storage (per GB)	**Reduced redundancy storage** (per GB)	**Glacier storage** (per GB)
First 1 (TB / month)	$0.095	$0.076	$0.010
Next 49 (TB / month)	$0.080	$0.064	$0.010
Next 450 (TB / month)	$0.070	$0.056	$0.010
Next 500 (TB / month)	$0.065	$0.052	$0.010
Next 4000 (TB / month)	$0.060	$0.048	$0.010
Next 5000 (TB / month)	$0.055	$0.037	$0.010

transforms the commonly large fixed costs into much smaller variable costs. They also remove the need to buy safety net capacity to handle periodic traffic spikes;

Reserved Instances: Reserved Instances give the user the option to make a one-time payment for each instance it wants to reserve and in turn receive a significant discount on the hourly charge for that instance. There are three Reserved Instance types: *light*, *medium*, and *heavy utilization reserved*, which enable the user to balance the amount it pays upfront with effective hourly prices. A *Reserved Instance Marketplace* is also available, which provides users with the opportunity to sell the instances if their needs change (for example, want to move instances to a new AWS region, change to a new instance type, or sell capacity for projects that end before the reservation term expires);

Spot Instances: Spot Instances allow users to bid on unused EC2 capacity and run those instances for as long as their bid exceeds the current spot price. The spot price changes periodically based on supply and demand, and the user whose bids meet or exceed the price gains access to the available instances. If the user has flexibility in when the applications can run, using spot instances can significantly lower the costs.

An EC2 user has the choice of multiple instance types, operating systems, and software packages. It can select a configuration of memory, CPU, instance storage, and the optimal boot partition size that is optimal for specific choice of applications and operating systems, e.g., Linux distributions or Microsoft Windows Server. The user can also increase or decrease capacity within minutes, and has complete control of the instances with root access. Moreover, to achieve reliability, replacement instances can be rapidly and predictably commissioned. For each region, the current Service Level Agreement commitment is 99.95 % availability.

The instance creation and configuration can be done through simple web service interfaces, as illustrated in Fig. 19.4. The user first selects a preconfigured *Amazon Machine Image* (AMI) template to boot up and run immediately (Step 1); or create an AMI containing the applications, libraries, data, and associated configuration settings. The user then chooses the expected instance type(s) (Steps 2 and 3), attaches storage, and set up network and security requirements, and starts, terminates, and

monitors the instances using the web service APIs or a variety of management tools. Security and network access can also be configured on the instance.

Designed for use with other AWS modules, EC2 works seamlessly in conjunction with Amazon S3, and such other Amazon services as Relational Database Service (RDS), SimpleDB, and Simple Queue Service (SQS) to provide a complete solution for computing, query processing, and storage across a wide range of applications, as Fig. 19.5 illustrates.

Specifically, an EC2 instance can be attached with a storage from the *Amazon Elastic Block Store* (EBS), which provides block level storage volumes from 1 GB to 1 TB. EBS provides the ability to create point-in-time snapshots of volumes, which can then be stored in S3 for long-term durability, as Fig. 19.6 illustrates.

19.2 Multimedia Cloud Computing

For multimedia applications and services over the Internet, there are strong demands for cloud computing, due to the massive storage and computation required for serving millions of wired Internet or mobile network users. In this new *multimedia cloud computing* paradigm [3], users can store and process their multimedia data in the cloud in a distributed manner, eliminating full installation of the media application software on local computers or devices.

Multimedia cloud computing shares many common characteristics with general-purpose cloud computing. Yet, multimedia services are highly heterogeneous. There exist a wide array of types of media and associated services, such as voice over IP, video conferencing, photo sharing and editing, image search, and image-based rendering, to name but a few; the cloud should support different types of media and their services for massive user bases simultaneously. Besides service heterogeneity, different types of devices, such as smart TVs, PCs, and smartphones, have different capabilities for multimedia processing. The cloud should have adaptive capability to fit these types of devices, in terms of CPU and GPU (Graphics Processing Unit), display, memory, and storage.

The multimedia cloud should also provide QoS provisioning to meet their distinct QoS requirements. There are two ways of providing QoS provisioning for multimedia: one is to add QoS to the current cloud-computing infrastructure and the other is to add a QoS middleware between the cloud infrastructure and the multimedia applications. The former focuses on the design and improvement within the cloud infrastructure. The latter focuses on improving cloud QoS in the middle layers, such as QoS in the transport layer and QoS mapping between the cloud infrastructure and media applications.

In summary, the heavy demands of multimedia data access, processing, and transmission would create bottlenecks in a general-purpose cloud. Today's cloud design has largely focused on allocating computing and storage resources through utility-like mechanisms, while QoS requirement in terms of bandwidth, delay, and jitter have yet to be addressed. To realize multimedia cloud computing, a synergy between the

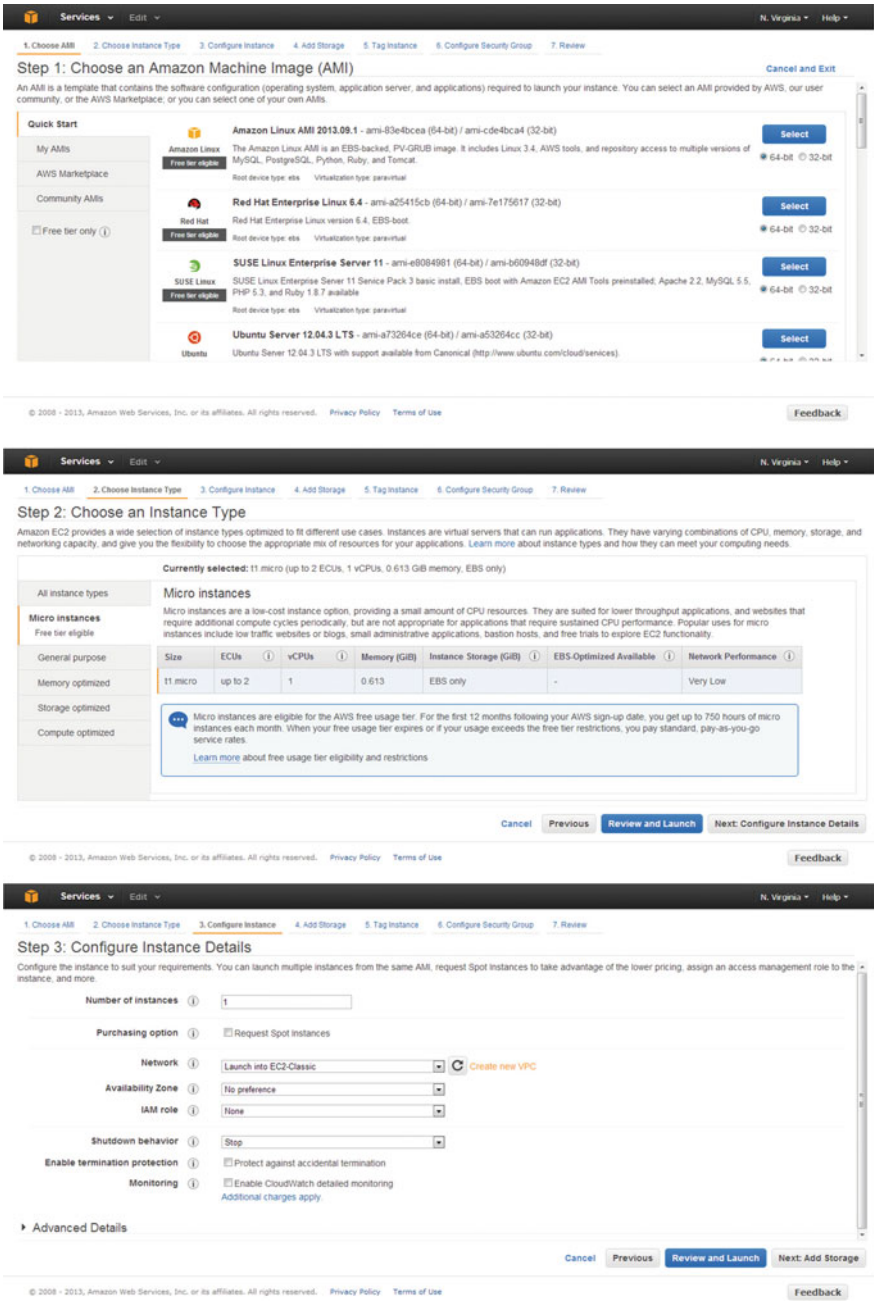

Fig. 19.4 Key steps in creating an Amazon EC2 instance

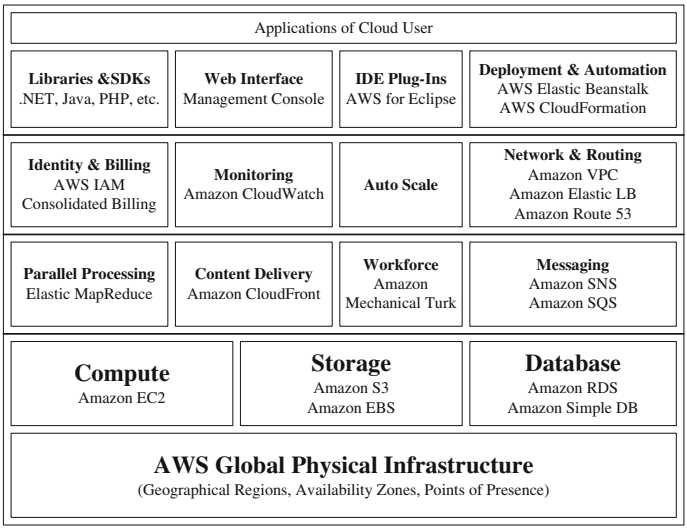

Fig. 19.5 Relations among the different components in Amazon Web Service (AWS)

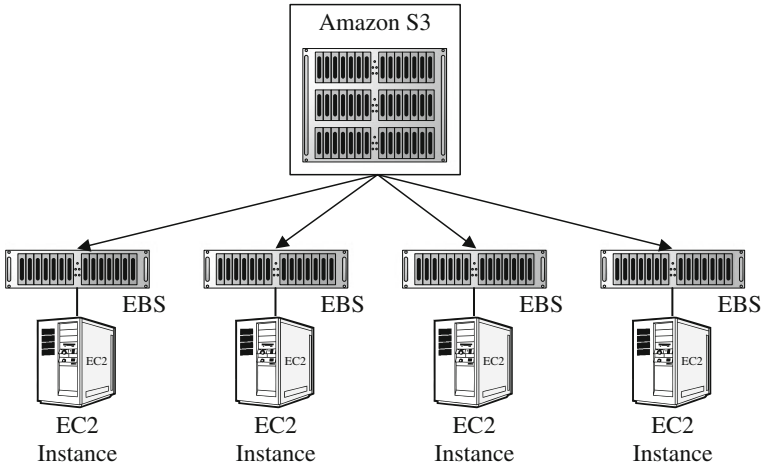

Fig. 19.6 The relations among Amazon S3, EC2, and the Elastic Block Store (EBS). EBS offers storage volumes from 1 GB to 1 TB that can be mounted as devices for EC2 instances, and the persistent storage is enabled by S3

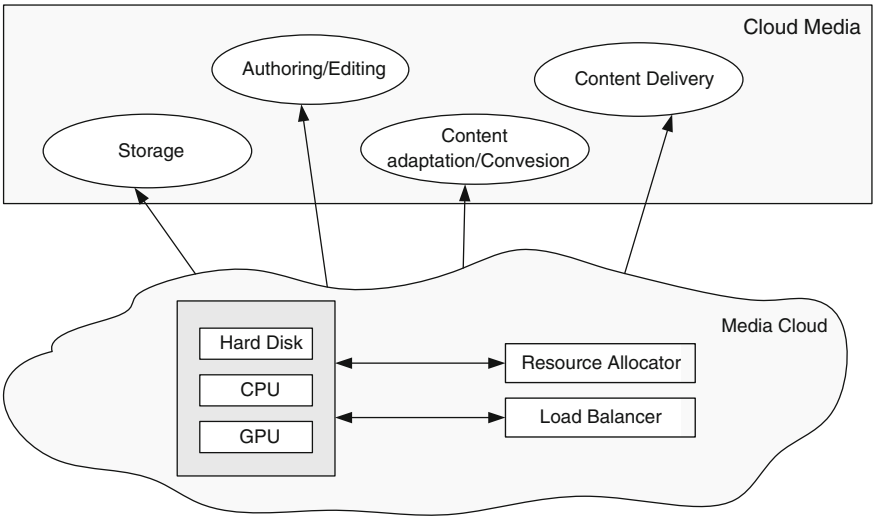

Fig. 19.7 Modules and their relations in multimedia cloud computing

cloud and multimedia services becomes necessary; that is, *multimedia-aware cloud computing* with enhanced QoS support for multimedia applications and services, and *cloud-aware multimedia* that facilitates content storage, processing, adaptation, rendering, in the cloud with optimized resource utilization [3], as Fig. 19.7 illustrates.

19.3 Cloud-Assisted Media Sharing

We first consider the use of the cloud for media sharing services. As we have seen in the previous chapters, representative media sharing services such as YouTube are developing extremely fast. In general, it is difficult if not impossible to predict the impact and the development of these new services in advance. The provision of resources is thus a great challenge, because any service with novel ideas, advanced techniques, and smart marketing strategies is possible to grow to the similar scale as YouTube. Yet there is a high possibility to fail, losing revenue, and even being shut down.

Developers face a dilemma at the early stage of media sharing services. On one hand, to provision large enough resources at the beginning is costly and risky, and if the service is not as popular as expected to gain enough revenue, the resources would be wasted. On the other hand, starting the service small usually comes with scalability issues. New features and increasing user base will put high pressure on the insufficient infrastructure, which downgrades the quality of service.

The cloud, which offers reliable, elastic, and cost-effective resource provisioning with "pay-as-you-go" service, is clearly an elegant solution here, allowing designers to start a service small but easy to scale large. Besides starting a service from the

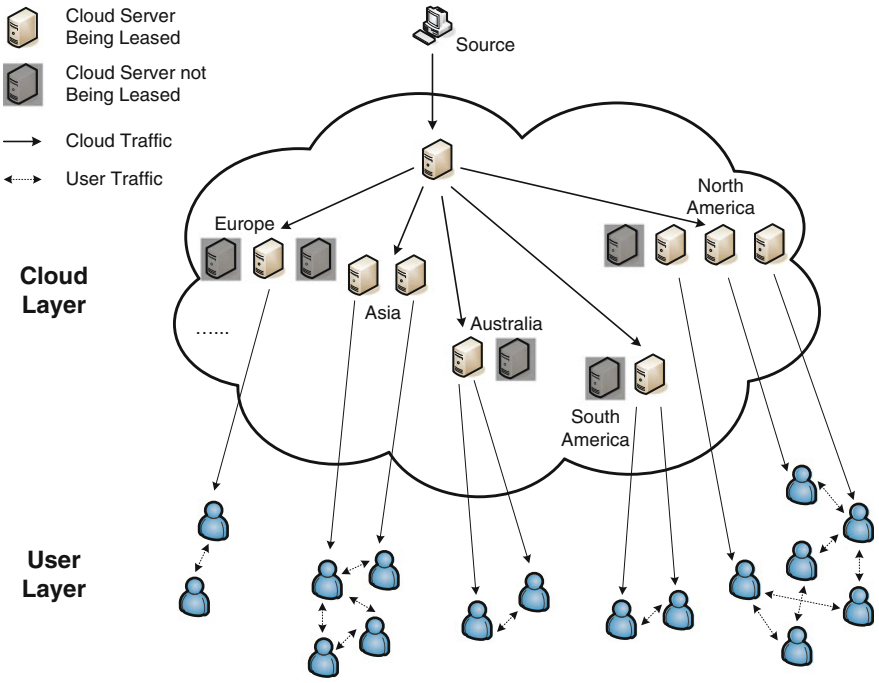

Fig. 19.8 A generic framework for migrating live media streaming service to the cloud

cloud, a migration that moves the contents to the cloud is beneficial as well for existing media services, in the presence of scalability challenges.

Sharing is an integral part of cloud service. The request of easy and scalable sharing is the main reason that multimedia contents now occupy a large portion of cloud storage space. The always-on and centralized data centers can make one-to-many sharing highly efficient and synchronous—uploaded content can be readily shared to a large population instantly or lately. Sharing through a cloud could also offer better QoS given that the connections with data centers are generally good, not to mention the firewall and NAT (Network Address Translation) traversal problems commonly encountered in peer-to-peer sharing.

Figure 19.8 shows a generic framework that facilitates the migration of existing live media streaming services to a cloud-assisted solution. It is divided into two layers, namely, *Cloud Layer* and *User Layer*. The Cloud Layer consists of the live media source and dynamically leased cloud servers. Upon receiving a user's subscription request, the Cloud Layer will redirect this user to a properly selected cloud server. Such a redirection, however, is transparent to the user, i.e., the whole Cloud Layer is deemed to be a single source server from a user's perspective. Since the user demands change over time, which are also location-dependent, the Cloud Layer will accordingly adjust the amount and location distribution of the leased servers. Intuitively, it will lease more server resources upon demand increase during peak

times, and terminate leases upon decrease. The implementation of the User Layer can be flexible. They can be individual users purely relying on the Cloud Layer, or served by peer-to-peer or a CDN infrastructure, but seeking for extra assistance from the cloud during load surges. In other words, it can smoothly migrate diverse existing live streaming systems to the cloud.

There are however a number of critical theoretical and practical issues to be addressed in this generic framework. Though cloud services are improving, given the hardware, software, and network limits, latencies in resource provisioning remain exist, e.g., to start or terminate a virtual machine can take a few minutes in the current Amazon EC2 implementation. While such latencies have been gradually reduced with improved cloud design, they can hardly eliminated. Therefore, the system must well predict when to lease new servers to meet the changing demands and when to terminate a server to minimize the lease costs. This can be done by a demand forecast algorithm for cloud users [4].

19.3.1 Impact of Globalization

The larger, dynamic, and nonuniform client population further aggravates the problem. To make it even worse, today's media sharing services have become highly globalized, with subscribers from all over the world. Such a globalization makes user behaviors and demands even more diverse and dynamic. Consider the user demand distribution of PPTV, a popular live media streaming systems with multi-million subscribers [5, 6]. Figure 19.9 shows the distribution of two representative channels (CCTV3 and DragonBall) during one day. It is easy to see that they had attracted users from all over the world, and the peak time therefore shifted from region to region, depending on the timezone. For example, on the CCTV3 channel, the peak time of North America was around 20:00, while for Asian users, it was around 8:00. During the period 12:00–20:00, Asian users had very low demands, while European users generated most of their demands and the North American users also had moderate demands. Similar observations can also be found from the DragonBall channel, despite the fact that the streaming contents delivered on the two channels were completely different.

In this context, the cloud should be combined with the Content Distribution Network (CDN) solution, so as to serve the users with geo-distributed servers. This in fact a general trend in today's cloud development beyond highly centralized data centers. One example is Amazon's CloudFront, a cloud-based CDN that is integrated in AWS. Using a network of edge locations around the world, CloudFront caches copies of static content close to viewers, lowering latency when they download objects and offering high, sustained data transfer rates needed to deliver large popular objects to end users at scale. The requests for dynamic content can be carried back to the origin servers running in AWS, e.g., S3, over optimized network paths (see Fig. 19.10). These network paths are constantly monitored by Amazon, and the connections from CloudFront edge locations to the origin can be reused to serve dynamic content with the best possible performance.

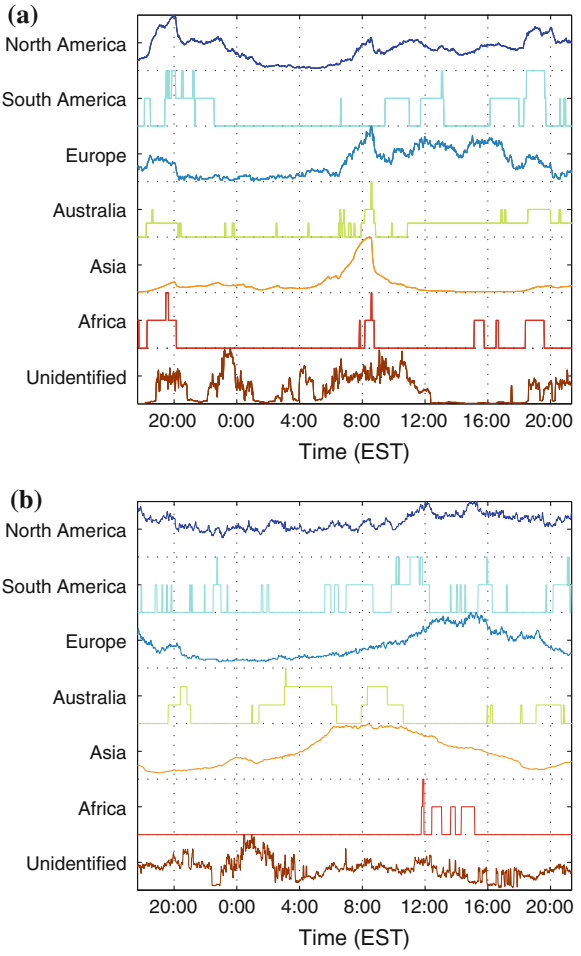

Fig. 19.9 An illustration of the user demand distributions and variations of a popular live media streaming system (PPTV) on its two typical channels (CCTV3 and DragonBall) during one day. For ease of comparison, the user demands have been normalized by the corresponding maximum demand of each day. The time shown on x-axis is based on EST. **a** CCTV-3, **b** DragonBall

19.3.2 Case Study: Netflix

One of the most successful migration of media sharing applications to the cloud is Netflix, which now takes up a third of US download Internet traffic during peak traffic hours. Established in 1997, Netflix began to move away from its original core business model of mailing DVDs by introducing video-on-demand via the Internet in early 2007. The original Netflix digital video distribution was based on a few large Oracle servers with a Java front end, with DVD subscription being the main business. Later in 2008, it had suffered from storage data corruption bugs that took service down. The rapidly increased scale creates another challenge, which however can hardly be predicted when building private server clusters, not to mention the staff and skills needed to run a large and high-growth rate data center infrastructure.

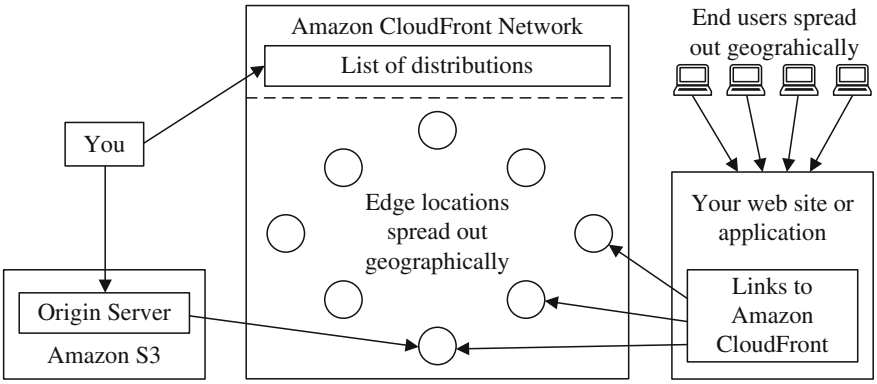

Fig. 19.10 An illustration of the CloudFront service, where a number of edge servers are geographically distributed, serving nearby cloud users

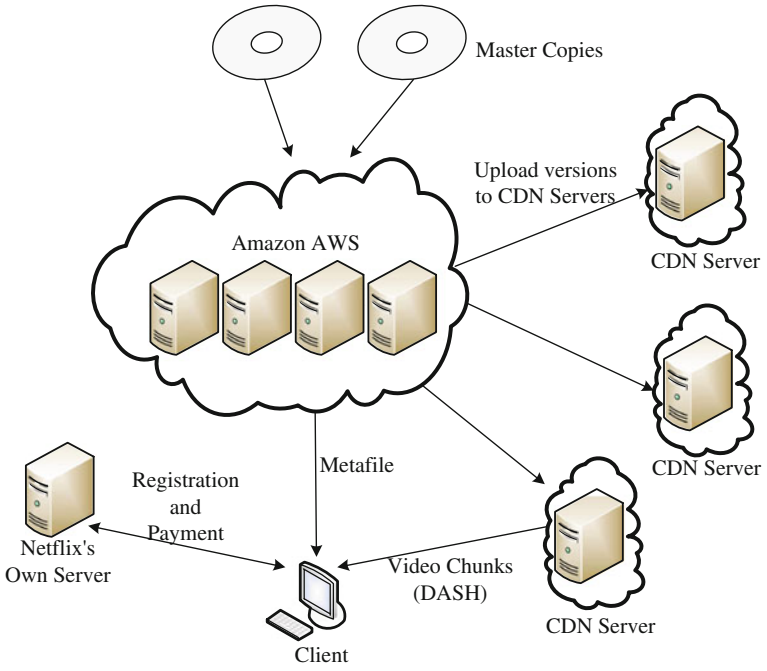

Fig. 19.11 The cloud-based Netflix architecture

Since 2009, Netflix started using Amazon's AWS for part of its services, and moved its entire technology infrastructure to AWS in 2012.

To support the combination of huge traffic and unpredictable demand bursts, Netflix has developed a global video distribution system using the AWS cloud. Figure 19.11 shows the architectural view of the cloud-based Netflix system, which includes the following key modules:

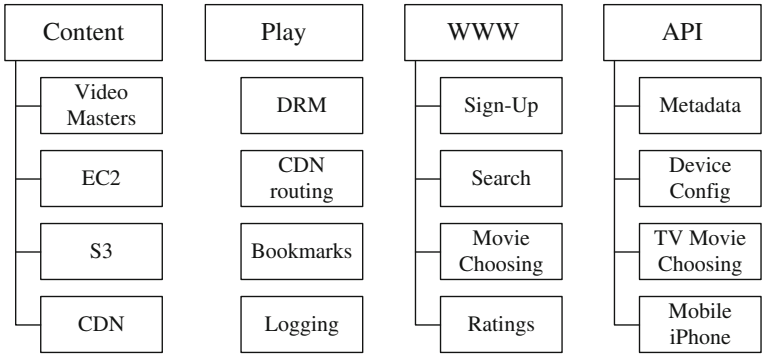

Fig. 19.12 The services integrated in Netflix and the cloud modules for them

1. **Content Conversion**. Netflix purchases master copies of digital films from movie studios and, using the powerful EC2 cloud machines, converts them to over 50 different versions with different video resolutions and audio quality, targeting a diverse array of client video players running on desktop computers, smartphones, and even DVD players or game consoles connected to television;
2. **Content Storage**. The master copies and the many converted copies are stored in S3. In total, Netflix has over 1 petabyte of data stored on Amazon;
3. **Content Distribution**. To serve worldwide users, the data are sent to content delivery networks (including Akamai, Limelight, and Level 3) that feed the content to local ISPs. With the versions of different formats and bit rates, Dynamic Adaptive Streaming over HTTP (DASH) is available for streaming to end users.

All of these services are distributed across three AWS availability zones. Netflix itself only maintains a minimum hardware infrastructure for user registration and credit card payment. Figure 19.12 shows different functional modules in Netflix and their relations with the cloud and CDNs.

Using Amazon's cloud and the CDNs, Netflix can react better and faster to user demand with every increasing scale. It pays by bytes for computation, bandwidth and storage resources so that the long-term costs become much lower than those with over-provisioning in self-owned servers. Such costs, without the cloud, can be prohibitively high for dedicated content providers.

19.4 Computation Offloading for Multimedia Services

Besides storage, computation is another rich resource offered by the cloud. Many computation-intensive tasks can now be migrated to the cloud, and the users do not have to maintain the ultra expensive high-performance servers or server clusters but just pay for the cost in an on-demand fashion.

Such *computation offloading* effectively expands the usability of local devices beyond their physical limits, which is particularly attractive for mobile devices [7–9]. Today's smartphones and tablets are increasingly penetrating into people's everyday life as efficient and convenient tools for communication and entertainment. The touch screen and all kinds of sensors provide even richer user experiences than desktop PCs do. Despite the fast development of such key components as CPU, GPU, memory, and wireless access technologies, and the effort toward unifying handheld and desktop computers, it remains widely agreed that mobile terminals will not completely replace laptop and desktop computers in the near future. Migrating popular PC software to mobile platforms or developing similar substitutes for them is still confined by their limited computation capability as well as the uniqueness of operating systems and hardware architectures. To make it even worse, battery, as the only power source of most mobile terminals, has seen relatively slow improvement in the past decade, which has become a major impediment in providing reliable and sophisticated mobile applications to meet user demands.

Combining the strength of the cloud and the convenience of mobile terminals thus becomes a promising route. An example is Apple's Siri service—after a piece of voice is recorded by an iPhone, a local recognizer will conduct speech recognition and decide whether to resort to the back end cloud to make an appropriate response. Other examples include MAUI [10] and CloneCloud [11]. The former enables fine-grained energy-aware offloading of mobile codes to a cloud based on the history of energy consumption. It achieves maximum energy savings of 90, 45, and 27 %, and maximum performance speedups of roughly 9.5, 1.5, and 2.5 for face recognition, chess, and video game, respectively. CloneCloud uses function inputs and an offline model of runtime costs to dynamically partition applications between a weak device and the cloud. It reports speedups of 14.05, 21.2, and 12.43 for virus scanning, image search, and behavior profiling, respectively.

19.4.1 Requirements for Computation Offloading

Assembling local resources and remote clouds organically to make offloading transparent to end users requires nontrivial effort [8, 9].

First, *motivation for offloading*. In the very beginning, the major motivation to offload should be determined, to save energy locally, to improve computation performance, or both. This will serve as a guideline in the high-level design.

Second, *gain of offloading*. To understand the potential gain after offloading, a profiling or breakdown analysis of the application is needed to see whether the application can benefit from offloading. There is no incentive to resort to the cloud for a job that can be easily and efficiently executed locally.

Third, *decision of offloading*. The offloading decision can be made statically or dynamically. For static offloading, the related parameters need to be accurately estimated in advance and the offloading strategy needs to be decided when developing the application. On the contrary, dynamic offloading monitors the runtime conditions and makes decisions accordingly, at the expense of higher overhead.

For multimedia applications, QoS requirements is also an important consideration. The critical QoS requirements (e.g., latency, image/video quality, computation accuracy) need to be respected so that offloading will not influence user experience. A simple solution for computation offloading is to move the whole computation engine to the remote cloud, then upload all the data required for computation to the cloud and download the computed results. While this is common in many implementations, complex enterprise applications are typically composed of multiple service components, which can hardly be migrated to the cloud in one piece and instantaneously.

This is further complicated with the wireless communications in mobile terminals. As compared to their wired counterparts, the mobile terminals are generally more resource-constrained; in particular, the wireless communication capacity and the battery capacity are their inherent bottlenecks [12, 13]. The energy trade-offs heavily depend on the workload characteristics, data communication patterns, and technologies used [14], all of which need to be carefully addressed. As such, offloading the whole computation module of an application to the remote cloud is not necessary, nor effective, if the data volume is large. This may not be a severe problem for users with high-speed wired network connection, but can dramatically reduce the benefit for mobile users.

19.4.2 Service Partitioning for Video Coding

Consider video encoding/compression, an essential task in a broad spectrum of mobile multimedia applications. A local user uses his/her mobile terminal to capture video in real-time, expecting to encode the video and then stream it to others in real-time as well. Directly uploading the raw video without efficient encoding inevitably leads to high bandwidth cost and large transmission energy consumption. On the other hand, video encoding often incurs heavy computation, which results in high energy consumption as well. For example, to encode a video of 5 seconds (30 frames per second with resolution of 176×144 and pixel depth of 8 bits) using an H.264 encoder needs almost 1×10^{10} CPU cycles in total [15], or 2×10^9 CPU cycles per second on average, which means that a 2 GHz CPU is required for real-time encoding. Considering that the newest smartphones and tablets are equipped with high-definition cameras, the CPU workload can be 5–10 times higher than that in the above example.

Offloading the whole video compression task to the cloud, however, is not practical because it is identical to directly uploading the raw video data. The wireless transmission can be either too costly or simply impossible with limited bandwidth. It is known that motion-estimation is the most computation-intensive module, accounting for almost 90 % of the computation. This module obviously should be the focus of offloading.

Yet it is not simple to decouple motion estimation from others given data dependency in coding, i.e., motion estimation of a frame depends on the data of the previous reference frame. A mobile terminal should upload the current video frame and the reference frame to the cloud for estimation and then download the estimated *Motion*

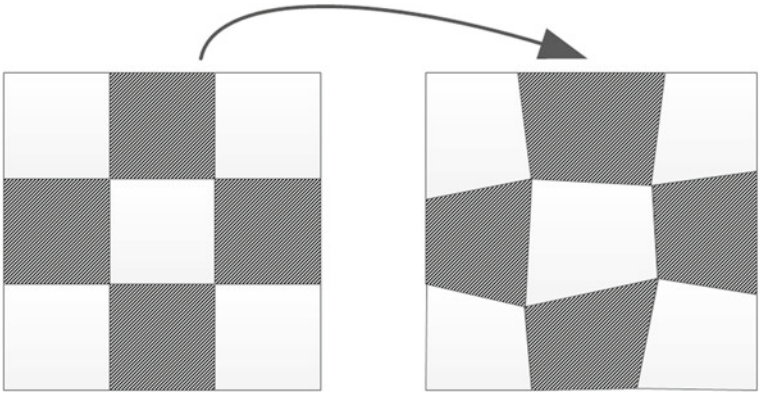

Fig. 19.13 Mesh-based motion estimation between two frames, using a regular mesh for the predictive frame

Vectors (MVs) from the cloud and complete the remaining video encoding steps (e.g., DCT and entropy coding). While the MVs are of small volume, uploading the current frames essentially makes no difference as compared to upload the raw video.

19.4.3 Case Study: Cloud-Assisted Motion Estimation

It is necessary to ensure that a minimum amount of data (not all the reference frame data) are to be uploaded to the cloud and yet allow estimation to be done accurately. Cloud-Assisted Motion Estimation (CAME) [16] addresses this issue by using *mesh-based motion estimation*, which partitions a frame into a coarse-grained mesh sketch and estimates one MV for each mesh node [17].

Regular triangular or rectangular meshes [18] have been commonly used given their simplicity, and both encoder and decoder can agree upon a mesh structure in advance. Unlike standard mesh-based motion estimation (see [17] for details), CAME applies a reversed mesh node selection and motion estimation, in which the mesh nodes are sampled on the P-frames and the MVs are calculated from the mesh nodes and the reference frame. As illustrated in Fig. 19.13, a set of regular mesh nodes are sampled on the P-frames of the macro block (MB). Only the sampled mesh nodes and the reference frame are uploaded to the cloud. The MVs are then calculated on the cloud, and the result, to be sent back to the mobile terminal, is a set of MVs that describe the movements of the mesh nodes. Using this design, the most computation-intensive part of mesh-based motion estimation is offloaded to the cloud, while the local motion estimation for individual macro-blocks within the mesh becomes much simpler. Compared to the standard mesh-based motion estimation, CAME loses the advantage of tracking the same set of mesh nodes over successive P-frames. It, however, saves more data transmission.

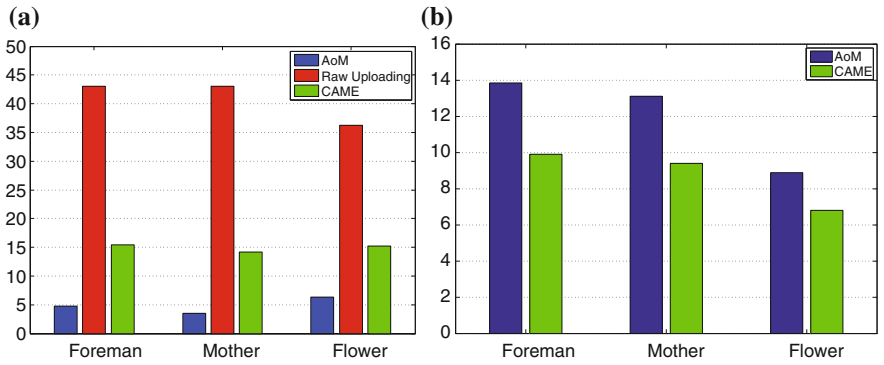

Fig. 19.14 Simulation results. **a** Total transmission volume (MB), **b** Total energy consumption in CPU cycles (billion cycle)

Figure 19.14a compares the total amount of transmitted data for three standard video sequences (Foreman, Mother, and Flower) with two baselines—AoM (All on Mobile), which executes the entire video encoding on mobile terminal, and raw uploading, which simply uploads all the data to the cloud for compression. For fair comparison, the transmission energy is converted into equivalent CPU cycles. Though Flower's original video size is smallest, both AoM and CAME incur the highest transmission cost as compared with other two videos, because Flower has higher spatial details. It is not surprising that the transmission cost of AoM is the lowest among all three and raw uploading has the largest cost. Compared to AoM, CAME introduces more transmission because of the extra data transmission for mesh node uploading and mesh motion vectors downloading. On the other hand, compared to raw uploading, the CAME method still saves approximately 60 % on total data transmission.

Although CAME consumes more energy on transmission than AoM does, it saves on total energy consumption through offloading the most computation-intensive task, motion estimation, to cloud servers. It spends nearly 40 % less energy on computation than AoM. Further, Fig. 19.14b confirms the expectation that CAME can achieve up to 30 % total energy savings on video encoding and transmission as compared to AoM.

In summary, although it is very tempting and promising to leverage the much cheaper and more powerful resources on the cloud, the interaction between a mobile terminal and the cloud needs careful examination to avoid excessive transmission overhead. As such, partitioning tailored to specific applications is often expected, as the example of motion estimation shows. The trade-off between the energy for computation and that for transmission can be found in many other applications that rely on computation offloading to extend the battery lifetime.

The closed loop design with the remote cloud also introduces extra delays between mobile user and the cloud. As we will see in the next section, such extra delays are generally acceptable in practice, even for interactive applications.

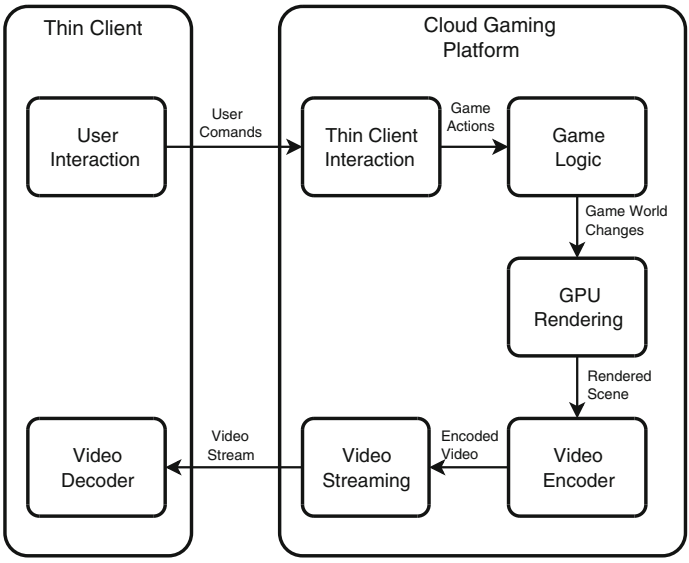

Fig. 19.15 A generic framework of cloud gaming

19.5 Interactive Cloud Gaming

Recently, advances in cloud technology have expanded to allow offloading not only of traditional computation but also of such more complex tasks as high definition 3D rendering, which turns the idea of *Cloud Gaming* into a reality [19, 20]. Cloud gaming, in its simplest form, renders an interactive gaming application remotely in the cloud and streams the scenes as a video sequence back over the Internet to the player. A cloud gaming player interacts with the application through a *thin client*, which is responsible for displaying the video from the cloud rendering server as well as collecting the player's commands and sending the interactions back to the cloud. Figure 19.15 shows a high-level architectural view of such a cloud gaming system with thin clients and cloud-based rendering.

Cloud gaming can bring great benefits by expanding the user base to the vast number of less-powerful devices that support thin clients only, particularly smartphones and tablets. As such, mobile users can enjoy high-quality video games without performing the computation-intensive image rendering locally. For example, the recommended system configuration for *Battlefield 3*, a highly popular first person shooter game, is a quad-core CPU, 4 GB RAM, 20 GB storage space, and a graphics card with at least 1 GB RAM (e.g., NVIDIA GEFORCE GTX 560 or ATI RADEON 6950), which alone costs more than $500. Even the newest tablets can hardly meet the minimum system requirements that need a dual-core CPU over 2.4 GHz, 2 GB RAM, and a graphics card with 512 MB RAM, not to mention smartphones of which the hardware is limited by their smaller size and thermal control. Furthermore, mobile devices have different hardware/software architecture from PCs, e.g., ARM rather

than x86 for CPU, lower memory frequency and bandwidth, and limited battery capacities. As such, the traditional console game model is not always feasible for such devices, which in turn become targets for cloud gaming. It also masks the discrepancy among different operating systems through such standard web development tools as HTML5, Flash, and JavaScript. It further reduces customer support costs since the computational hardware is now under the cloud gaming provider's full control, and offers better Digital Rights Management (DRM) since the codes are not directly executed on a customer's local device.

19.5.1 Issues and Challenges of Cloud Gaming

As shown in Fig. 19.15, in cloud gaming, a player's commands must be sent over the Internet from its thin client to the cloud gaming platform. Once the commands reach the remote cloud, they are converted into appropriate in-game actions, which are interpreted by the game logic into changes in the game world. The game world changes are processed by the cloud system's GPU into a rendered scene. The rendered scene is then compressed by the video encoder, and sent to a video streaming module, which delivers the video stream back to the thin client. Finally, the thin client decodes the video and displays the video frames to the player.

To ensure interactivity, all of these serial operations must happen in the order of milliseconds. Intuitively, this amount of time, which is defined as the *interaction delay*, must be kept as short as possible in order to provide a rich experience to the cloud game players. There are however tradeoffs: the shorter the player's tolerance for interaction delay, the less time the system has to perform such critical operations as scene rendering and video compression. Also, the lower this time threshold is, the more likely a higher network latency can negatively affect a player's experience of interaction.

Interaction Delay Tolerance
Studies on traditional gaming systems have found that different styles of games have different thresholds for maximum tolerable delay [21]. Table 19.2 summarizes the maximum delay that an average player can tolerate before the *Quality-of-Experience* (QoE) [19] begins to degrade. As a general rule, the games that are played in the first person perspective, such as the shooter game `Counter Strike`, become noticeably less playable when actions are delayed by as little as 100 ms. This low delay tolerance is because such first person games tend to be action-based, and players with a higher delay tend to have a disadvantage [22]. In particular, the outcome of definitive game changing actions such as who "pulled the trigger" first can be extremely sensitive to the delay in an action-based *First Person Shooter* (FPS) game.

Third person games, such as *Role Playing Games* (RPG), and many massively multiplayer games, such as `World of Warcraft`, can often have a higher delay tolerance of up to 500 ms. This is because a player's commands in such games, e.g., use item, cast spell, or heal character, are generally executed by the player's avatar; there is often an invocation phase, such as chanting magic words before a spell is cast, and hence the player does not expect the action to be instantaneous. The actions

Table 19.2 Delay tolerance in traditional gaming

Example game type	Perspective	Delay threshold (ms)
First person shooter (FPS)	First person	100
Role playing game (RPG)	Third-person	500
Real-time strategy (RTS)	Omnipresent	1000

must still be registered in a timely manner, since the player can become frustrated if the interaction delay causes them a negative outcome, e.g., they healed before an enemy attack but still died because their commands were not registered by the game in time.

The last category of games are those played in an "omnipresent" view, i.e., a top-down view looking at many controllable entities. Examples are *Real-Time Strategy* (RTS) games like `Star Craft` and such simulation games as `The Sims`. Delays of up to 1000 ms can be acceptable to these styles of games since the player often controls many entities and issues many individual commands, which often take seconds or even minutes to complete. In a typical RTS game, a delay of up to 1000 ms for a build unit action that takes over a minute will hardly be noticed by the player.

Although there is much similarity between interaction delay tolerance for traditional gaming and cloud gaming, it is useful to stress the following critical distinctions. First, traditionally, the interaction delay was only an issue for multiplayer online gaming systems, and was generally not considered for single player games. Cloud gaming drastically changes this: now all games are being rendered remotely and streamed back to the player's thin client. As such, we must be concerned with interaction delay even for a single player game. Also, traditional online gaming systems often hide the effects of interaction delay by rendering the action on a player's local system before it ever reaches the gaming server. For example, a player may instruct the avatar to move and it immediately begins the movement locally; however, the gaming server may not receive the update on the position for several milliseconds. Since cloud gaming offloads its rendering to the cloud, the thin client no longer has the ability to hide the interaction delay from the player. Such visual cues as mouse cursor movement can be delayed by up to 1000 ms, making it impractical to expect the player will be able to tolerate the same interaction delays in cloud gaming as they do in traditional gaming systems. The maximum interaction delay for all games hosted in a cloud gaming context should be no more 200 ms. Other games, specifically such action-based games as first person shooters likely require less than 100 ms interaction delay in order not to affect the players's QoE.

Video Streaming and Encoding

Cloud gaming's video streaming requirements are quite similar to live video streaming. Both cloud gaming and live video streaming must quickly encode/compress incoming video and distribute it to end users. In both cases, only a small set of the most recent video frames are of interest, and there is no need or possibility to access future frames before they are produced, implying encoding must be done with respect to very few frames.

Yet conventional live video streaming and cloud gaming have important differences. First, compared to live video streaming, cloud gaming has virtually no capacity to buffer video frames on the client side. This is because, when a player issues a command to the local thin client, the command must traverse the Internet to the cloud, be processed by the game logic, rendered by the processing unit, compressed by the video encoder and streamed back to the player. Given that this must all be done in under 100–200 ms, it is apparent that there is not much margin for a buffer. Live video streaming on the other hand can afford a buffer of hundreds of milliseconds or even a few seconds with very little loss to the QoE of the end user.

The sensitive real-time encoding needs of cloud gaming make the choice of video encoder of paramount importance for any cloud gaming provider. Currently, major cloud gaming providers, such as Gaikai and Onlive, use versions of the H.264/MPEG-4 AVC encoder. Gaikai uses a software based approach for encoding whereas Onlive is using specialized hardware to compress its cloud gaming video streams. In either case, the choice of the H.264/MPEG-4 AVC encoder is motivated by the fact that the encoder not only has a very high compression ratio but also that it can be configured to work well with stringent real-time demands.

19.5.2 Real-World Implementation

Onlive and Gaikai are two industrial pioneers of cloud gaming. Boosting resource-limited users to play the games that used to be exclusive for high-end PCs and gaming consoles, both of them have seen great success with multimillion user bases. The Sony's new Playstation 4 game console will also uses Gaikai's cloud platform.

Gaikai is implemented using two public clouds, namely Amazon EC2 and Limelight. Figure 19.16 offers a practical view of Gaikai's work flow. When a user selects a game on Gaikai (*Step1* in Fig. 19.16), an EC2 virtual machine will first deliver the Gaikai game client to the user (*Step2*). After that, it forwards the IP addresses of game proxies that are ready to run the selected games to the user (*Step3*). The user will then select one game proxy to run the game (*Step4*). The game proxy starts to run the game and the game screens will be streamed to the user via UDP (*Step5* and *Step6*). For multiplayer online games, these game proxies will also forward user operations to game servers (mostly deployed by the game companies) and send the related information/reactions back to the users (*Step7*).

Onlive's workflow is quite similar, but is implemented with a private cloud environment. Using public clouds enables lower implementation costs and higher scalability; yet a private cloud may offer better performance and customization that could fully unleash the potentials of cloud for gaming.

For a popular game, `Batman Arkham Asylum`, which is streamed at $1,280 \times 720$ pixels (720p) by Onlive, Table 19.3 illustrates the effect of Onlive's compression taken from a single frame of the opening sequence. As compared to the image quality from the local game console, the effect of compression is noticeable, especially when the amount of available bandwidth decreases. Although the image quality is fine with high bandwidth (10 Mbps), given today's limited Internet access bandwidth, there is still a room for cloud gaming to improve.

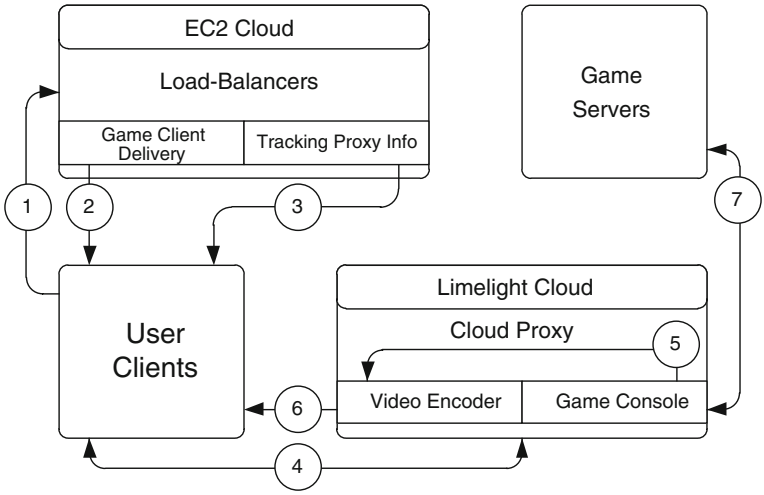

Fig. 19.16 The workflow of the Gaikai cloud gaming platform

Table 19.3 Image quality comparison

Measurement	PSNR (dB)	SSIM
Master image	n/a	n/a
Local capture	33.85	0.97
Onlive (10 Mbps)	26.58	0.94
Onlive (6 Mbps)	26.53	0.92
Onlive (3 Mbps)	26.03	0.89

The video frames are analyzed using two classical metrics, namely *Peak Signal-to-Noise Ratio* (PSNR) and *Structural Similarity (SSIM)* [23]. The SSIM method calculates the similarities in image structures (features) between the two video frames. As can be seen, the local capture scored a high PSNR and SSIM; however it is not perfect, indicating some difference in the recorded video and the master file. Much of this difference is likely due to slightly different brightness and color settings used by the internal video player in the Batman game engine. When the local capture is compared to Onlive running at any connection rate, it can be seen that there is large drop in terms of both PSNR and SSIM. Since PSNR and SSIM are not on a linear scale, the drops actually indicate a considerable degradation in image quality. Generally, a PSNR of 30 dB and above is considered good quality, however 25 dB and above is considered acceptable for mobile video streaming. Not surprisingly, as the amount of available bandwidth decreases the image quality begins to suffer considerable degradation as well.

Figure 19.17 shows the extra latency brought by Onlive. For example, Onlive (+20 ms) indicates that an extra 20 ms is added on the network delay, bringing the total to 50 ms. The locally rendered copy has an average interaction delay of approximately

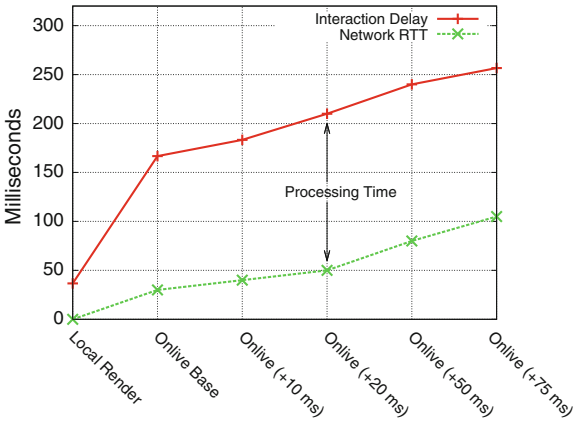

Fig. 19.17 Interaction delay in Onlive

37 ms, whereas the Onlive baseline takes approximately four times longer at 167 ms to register the same game action. As is expected, for higher network latencies, the interaction delay increases. Impressively, the Onlive system manages to keep its interaction delay below 200 ms in many of our tests. This indicates that, for many styles of games, Onlive could provide acceptable interaction delays. However, when the network latency exceeds 50 ms, the interaction delays may begin to hinder the users' experience. Also, even with the baseline latency of only 30 ms, the system could not provide an interaction delay of less than 100 ms, the expected threshold for first person shooters.

Table 19.4 further gives the detailed breakdown for interaction processing and cloud overhead. The processing time is the amount of interaction delay caused by the game logic, GPU rendering, video encoding, and etc; that is, it is the components of the interaction delay not explained by the network latency. The cloud overhead is the delay not caused by the core game logic or network latency, including the amount of delay caused by the video encoder and streaming system used in Onlive.

As can be seen, the cloud processing adds about 100–120 ms of delay in Onlive. This is better than earlier studies that show the cloud overhead is around 200 ms [24], suggesting that cloud gaming technology improves very fast. On the other hand, local rendering by the game console needs less than 37 ms. In other words, although the cloud, powered by data centers, is principally more powerful than any local console, the current implementation is not very efficient. To reach the optimal interaction delay threshold, better implementations in terms of game logic, video encoders, and streaming software remain expected for cloud gaming.

The interaction paths can be further prolonged in a large-scale multiuser game, where the globally distributed users are served by different cloud data centers. The cloud providers, however, are often served by better network connections (e.g., near to the backbone networks and higher bandwidth) or even dedicated high-speed networks. If the cloud servers are smartly assigned to user clients, the latency may not change much and is still acceptable.

Table 19.4 Processing time and cloud overhead

Measurement	Processing time (ms)	Cloud overhead (ms)
Local render	36.7	n/a
Onlive base	136.7	100.0
Onlive (+10 ms)	143.3	106.7
Onlive (+20 ms)	160.0	123.3
Onlive (+50 ms)	160.0	123.3
Onlive (+75 ms)	151.7	115.0

Cloud gaming is a rapidly evolving technology, with many exciting possibilities. Besides software and service providers, hardware manufacturers have also shown strong interests in cloud gaming, and have begun working on dedicated hardware solutions to address such prominent issues as fast and concurrent rendering and encoding of game scenes [25], which certainly sheds lights to the future of cloud gaming.

19.6 Further Exploration

Cloud computing remains a new field for both industry and research community. Many of the related materials can be found as white papers from such major cloud computing providers as Amazon, Google, and Microsoft.

19.7 Exercises

1. Discuss the relations and differences of the following systems: Cloud, Server Cluster, Content Distribution Network (CDN), and Datacenter.
2. Consider cloud-based video streaming and peer-to-peer video streaming.
   (a) For each of them, what are the pros and cons?
   (b) Discuss a possible solution that combines these two. Discuss the benefit of this hybrid design and its potential issues.
3. Consider the cloud-based Netflix Video-on-Demand (VoD) service.
   (a) Describe the respective roles of Amazon EC2 and S3 in Netflix.
   (b) Netflix uploads the master videos to the cloud for transcoding. Why doesn't Netflix transcode the videos locally and then upload them to the cloud?
   (c) Why does Netflix still need a CDN service beyond S3?
4. Is it always beneficial to offload computation to the cloud? List two application scenarios that simple computation offloading may not be cost-effective, and suggest possible solutions.

5. In this question, we try to quantify the cost savings of using cloud services. Without the cloud, a user has to purchase his or her own PC, say of price $\$X$. The value of the machine depreciates at a rate of $p\,\%$ per month, and when the value is below $V\,\%$, the machine is considered out-dated and the user has to purchase a new machine. On the other hand, using the cloud, the user doesn't have to buy his or her own machine, but leases from the cloud service provider with a monthly fee of $\$C$.
   (a) To make the cloud service cost-effective, how much should the provider set for the monthly lease fee $\$C$ for a comparable cloud machine instance?
   (b) Are there any other real-world costs that can be included in the model, associated either with local machine or with the cloud?
6. Considering the energy consumption of a task with local computing at a mobile terminal and with offloading to the cloud. We assume the task needs C CPU cycles for computation. Let M and S be the computing speeds, in CPU cycles per second, of the mobile terminal and of the cloud machine, respectively. The local computing at the mobile terminal has energy consumption of P_M watts, and incurs no data exchange over the wireless interface. For offloading to the cloud, D bytes of data are to be exchanged over the wireless interface. We assume that the network bandwidth is B bps and the energy consumption of the air interface during transmitting or receiving is P_T watts.
   (a) Assume that the CPU of the mobile terminal consumes no energy when it is idle, nor does the wireless interface of the terminal. What is the energy consumption of the mobile terminal if the task is executed locally, or offloaded to the cloud? Note that we don't consider the energy consumption in the cloud because the energy bottleneck of interest here is at the mobile terminal.
   (b) Under what condition does offloading to the cloud save energy?
   (c) What are other potential benefits with computation offloading, and under what conditions?
7. Besides the cost or energy savings, list two other advantages when using a cloud, as compared to building and maintaining a local infrastructure. Also list two disadvantages.
8. Consider cloud gaming, in which game scenes are rendered in the cloud and then streamed back to a thin client.
   (a) What are the benefits of using a cloud for gaming?
   (b) What types of games are most suitable for cloud gaming?
   (c) Discuss the requirements for live video streaming and those for cloud gaming. How are they similar? What special requirements of cloud gaming make it more difficult?
   (d) Suggest some solutions that can reduce the delay in cloud gaming.

References

1. M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
2. P. Mell, T. Grance, The nist definition of cloud computing. Technical Report Special Publication 800–145, National Institute of Standards and Technology (NIST) (2011)
3. W. Zhu, C. Luo, J. Wang, S. Li, Multimedia cloud computing. IEEE Signal Process. Mag. **28**(3), 59–69 (2011)
4. D. Niu, Z. Liu, B. Li, S. Zhao, Demand forecast and performance prediction in peer-assisted on-demand streaming systems. In *Proceedings of the IEEE INFOCOM Mini-Conference*, 2011
5. Y. Huang, T. Fu, D. Chiu, J. Lui, C. Huang, Challenges, design and analysis of a large-scale P2P-VoD system. In *Proceedings of the ACM SIGCOMM*, 2008
6. K. Xu, H. Li, J. Liu, W. Zhu, W. Wang, PPVA: a universal and transparent peer-to-peer accelerator for interactive online video sharing. In *Proceedings of the IEEE IWQoS*, 2010
7. F. Liu, P. Shu, H. Jin, L. Ding, D. Niu, B. Li, Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications. IEEE Wirel. Commun. **20**(3), 14–22 (2013)
8. X. Ma, Y. Zhao, L. Zhang, H. Wang, L. Peng, When mobile terminals meet the cloud: computation offloading as the bridge. IEEE Network **27**(5), 28–33 (2013)
9. K. Kumar, Y.-H. Lu, Cloud computing for mobile users: can offloading computation save energy? IEEE Comput **43**(4), 51–56 (2010)
10. E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services, MobiSys '10*, (ACM, New York, NY, USA, 2010), pp. 49–62
11. B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on Computer Systems, EuroSys '11*, (ACM, New York, NY, USA, 2011), pp. 301–314
12. K. Kumar, J. Liu, Y.-H. Lu , B. Bhargava, A survey of computation offloading for mobile systems. Mob. Networks Appl. **18**(1), 129–140 (2013)
13. H.T. Dinh, C. Lee, D. Niyato, P. Wang, A survey of mobile cloud computing: architecture, applications, and approaches. Wirel. Commun. Mob. Comput (in press)
14. A.P. Miettinen, J.K. Nurminen, Energy efficiency of mobile clients in cloud computing. In *Proceedings of the 2nd USENIX conference on Hot Topics in Cloud Computing, HotCloud'10*, (USENIX Association, Berkeley, CA, USA, 2010), pp. 4–4
15. N. Imran, B.-C. Seet, A.C.M. Fong, A comparative analysis of video codecs for multihop wireless video sensor networks. Multimedia Syst. **18**(5), 373–389, (2012)
16. Y. Zhao, L. Zhang, X. Ma, J. Liu, H. Jiang, CAME: cloud-assisted motion estimation for mobile video compression and transmission. In *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video, NOSSDAV '12*, (ACM, New York, NY, USA, 2012), pp. 95–100
17. Y. Wang, J. Ostermann, Y.-Q. Zhang, *Video Processing and Communications*, vol. 5 (Prentice Hall, Upper Saddle River, 2002)
18. M. Sayed, W. Badawy, A novel motion estimation method for mesh-based video motion tracking. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004 Proceedings (ICASSP'04)*, (IEEE, vol. 3, 2004), pp. iii–337
19. M. Jarschel, D. Schlosser, S. Scheuring, T. Hossfeld, An evaluation of qoe in cloud gaming based on subjective tests. In *Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pp. 330–335, 2011
20. R. Shea, J. Liu, E.C.-H. Ngai, Y. Cui, Cloud gaming: architecture and performance. IEEE Network **27**(4), 16–21 (2013)

21. M. Claypool, K. Claypool, Latency and player actions in online games. Commun. ACM **49**(11), 40–45 (2006)
22. M. Claypool, K. Claypool, Latency can kill: precision and deadline in online games. In *Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems, MMSys'10*, (ACM, New York, NY, USA, 2010), pp. 215–222
23. Z. Wang et al., Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
24. K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, C.-L. Lei, Measuring the latency of cloud gaming systems. In *Proceedings of the 19th ACM International Conference on Multimedia, MM '11*, pp. 1269–1272, 2011
25. Z. Zhao, K. Hwang, J. Villeta, Game cloud design with virtualized cpu/gpu servers and initial performance results. In *Proceedings of the 3rd Workshop on Scientific Cloud Computing Date, ScienceCloud '12*, (ACM, New York, NY, USA, 2012), pp. 23–30