In this section we discuss a cloud application related to data streaming [288]. Data streaming is the

name given to the transfer of data at a high rate with real-time constraints. Multimedia applications such

as music and video streaming, high-definition television (HDTV), scientific applications that process a

continuous stream of data collected by sensors, the continuous backup copying to a storage medium of

the data flow within a computer, and many other applications require the transfer of real-time data at

a high rate. For example, to support real-time human perception of the data, multimedia applications

have to make sure that enough data is being continuously received without any noticeable time lag.

We are concerned with the case when data streaming involves a multimedia application connected to

a service running on a computer cloud. The stream could originate from the cloud, as is the case of the

iCloud service provided by Apple, or could be directed toward the cloud, as in the case of a real-time

data collection and analysis system.

Data streaming involves three entities: the sender, a communication network, and a receiver. The

resources necessary to guarantee the timing constraints include CPU cycles and buffer space at the

sender and the receiver, as well as network bandwidth. Adaptive data streaming determines the data rate

based on the available resources. Lower data rates imply lower quality, but they reduce the demands for

system resources.

Adaptive data streaming is possible only if the application permits tradeoffs between quantity and

quality. Such tradeoffs are feasible for audio and video streaming, which allow lossy compression, but

are not acceptable for many applications that process a continuous stream of data collected by sensors.

Data streaming requires accurate information about all resources involved, and this implies that the

network bandwidth has to be constantly monitored; at the same time, the scheduling algorithms should

be coordinated with memory management to guarantee the timing constraints. Adaptive data streaming

poses additional constraints because the data flow is dynamic. Indeed, once we detect that the network

cannot accommodate the data rate required by an audio or video stream, we have to reduce the data

rate; thus, to convert to a lower quality audio or video. Data conversion can be done on the fly and, in

this case, the data flow on the cloud has to be changed.

Accommodating dynamic data flows with timing constraints is nontrivial; only about 18% of the top

100 global video Web sites use adaptive bit rate (ABR) technologies for streaming [336].

This application stores the music files in S3 buckets, and the audio service runs on the EC2 platform.

In EC2 each virtual machine functions as a virtual private server and is called an instance; an instance

specifies the maximum amount of resources available to an application, the interface for that instance,

and the cost per hour.

EC2 allows the import of virtual machine images from the user environment to an instance through

a facility called VM import. It also distributes automatically the incoming application traffic among

multiple instances using the elastic load-balancing facility. EC2 associates an elastic IP address with

an account; this mechanism allows a user to mask the failure of an instance and remap a public IP

address to any instance of the account, without the need to interact with the software support team.

Adaptive audio streaming involves a multi-objective optimization problem. We want to convert the

highest-quality audio file stored on the cloud to a resolution corresponding to the rate that can be

sustained by the available bandwidth; at the same time, we want to minimize the cost on the cloud site and minimize the buffer requirements for the mobile device to accommodate the transmission jitter.

Finally, we want to reduce to a minimum the startup time for the content delivery.

A first design decision is whether data streaming should only begin after the conversion from the

WAV to MP3 format has been completed or it should proceed concurrently with conversion – in other

words, start as soon as several MP3 frames have been generated. Another question is whether the

converted music file should be saved for later use or discarded.

To answer these questions, we experimented with conversion from the highest-quality audio files,

which require a 320 Kbps data rate, to lower-quality files corresponding to 192, 128, 64, 32, and finally

16 Kbps. If the conversion time is small and constant there is no justification for pipelining data

conversion and streaming, a strategy that complicates the processing flow on the cloud. It makes sense

to cache the converted copy for a limited period of time with the hope that it will be reused in the future.

Another design decision is how the two services should interact to optimize performance. Two

alternatives come to mind:

1. The audio service running on the EC2 platform requests the data file from the S3, converts it, and,

eventually, sends it back. The solution involves multiple delays and it is far from optimal.

2. Mount the S3 bucket as an EC2 drive. This solution reduces considerably the start-up time for audio

streaming.

The conversion from a high-quality audio file to a lower-quality, thus a lower-bit-rate, file is performed

using the LAME library.

The conversion time depends on the desired bitrate and the size of the original file. Tables 11.2, 11.3,

11.4, and 11.5 show the conversion time in seconds when the source MP3 files are of 320 Kbps and 192

Kbps, respectively. The sizes of the input files are also shown.

The platforms used for conversion are (a) the EC2 t1.micro server for the measurements reported

in Tables 11.2 and 11.3 and (b) the EC2 c1.medium for the measurements reported in Tables 11.4 and

11.5. The instances run the Ubuntu Linux operating system.

The results of our measurements when the instance is the t1.micro server exhibit a wide range of

conversion times, 13–80 seconds, for the large audio file of about 6.7 MB when we convert from 320

to 192 Kbps. A wide range, 13–64 seconds, is also observed for an audio file of about 4.5 MB when we convert from 320 to 128 Kbps. For poor-quality audio the file size is considerably smaller, about

0.56 MB, and the conversion time is constant and small, 4 seconds. Figure 11.15 shows the average

conversion time for the experiments summarized in Tables 11.2 and 11.3. It is somewhat surprising that the average conversion time is larger when the source file is smaller, as is the case when the target bit

rates are 64, 32, and 16 Kbps.

Figure 11.16 shows the average conversion time for the experiments summarized in Tables 11.4

and 11.5.

The results of our measurements when the instance runs on the EC2 c1.medium platform show

consistent and considerably lower conversion times; Figure 11.16 presents the average conversion time.

To understand the reasons for our results, we took a closer look at the two types of AWS EC2

instances, “micro” and “medium,” and their suitability for the adaptive data-streaming service. The

t1.micro supports bursty applications, with a high average-to-peak ratio for CPU cycles, e.g., transaction processing

systems. The Amazon Elastic Block Store (EBS) provides block-level storage volumes; the

“micro” instances are only EBS-backed.

The “medium” instances support compute-intensive application with a steady and relatively high

demand for CPU cycles. Our application is compute-intensive; thus, there should be no surprise that

our measurements for the EC2 c1.medium platform show consistent and considerably lower conversion

times.