# 3 An Architecture for Public and Open Submission Systems in the Cloud

## 3.1 Problem Addressed

The Brazilian Big Brother reality show is broadcasted by free-to-air TV network with an audience of more than eighty million people simultaneously. The idea behind this reality show is to portray the life of 16 random anonymous people while living together under the same roof, for a total period of three months. They are isolated from the outside world but are continuously monitored by television cameras. The housemates try to win a cash prize by avoiding periodic evictions from the house.

With technological advances the application process evolved from sending a videotape by postal mail to uploading a digital video using the Internet. Due to legal reasons, videos can not be hosted in websites such as YouTube or Vimeo. Applicants are allowed to send videos in the video format of their choice. These must be stored until the end of the selection process (three months). All the videos need to be transcoded to a standard format, so that the TV show's production team is spared from the hassle of having to deal with a plethora of video formats and different codecs.

The system should be able to receive a very large number of videos during the three-month application process. With the new digital process it is expected that more than 100,000 videos; about 60% of the total submission is uploaded during the last week before the deadline.

## 3.2 Research Relevance

Investment in infrastructure for high peak situations for a short period of time is usually a waste of money and resources as most of the time the resources will not be used. In what follows we will argue that Cloud Computing [ARMBRUST 2009] technology provides the necessary requirements in which to provide a viable solution. It provides the necessary infrastructure in which to develop submission applications in which both storage and processing needs can be dimensioned as needed. For this purpose we propose a general architecture for open, public submission systems – thus allowing up and down scalability rapidly responding to external factors.

The relevance of this demo is the use of Cloud Computing [VOGELS 2008] to solve a real world problem with a general-purpose architecture that could be re-used in different situations. This architecture provides the necessary flexibility to be used in a wide range of applications [MILLER 2008] that deals with large dataset processing, such as text corpus processing, audio recognition, and, as described in this demo, for mass video transcoding, and, when deployed in a Cloud platform, providing a dynamic and efficient resource usage, which might be a critical factor to business success.

## 3.3 Uniqueness of Design and Implementation

In this section, we describe the proposed architecture for large, user-generated content, file submission and processing systems using Cloud Computing. A few specific characteristics leverage the use of Cloud Computer architecture for this project in particular:

- Uncertainty in how much storage and processing capacity will be needed;
- Resources will be needed during the application and selection processes only. After this short period, all storage and computing resources would be idle;
- Few but extreme high peak situations where the infrastructure will need to scale up – 60% of the videos are expected to be sent in the last week;

The proposed architecture uses the cloud to store and process all this content, and to provide storage availability and scale resources as needed. All user content is received through a website where video files can be uploaded without restriction regarding the file extension, or video format/codec.

The demo is based on Amazon's Cloud Computing platform and we make use of the following services:

**Amazon S3** – Amazon Simple Storage Service is cloud-based persistent storage and operates independently from other Amazon services. It can be used to upload data in the cloud and pull it back out.

**Amazon EC2** – Amazon Elastic Compute Cloud is a web service [ZHANG 2004] that provides resizable compute capacity in the cloud. It provides an API for provisioning, managing, and deprovisioning virtual servers inside the Amazon cloud. It's the heart of the cloud and allows remote deployment of virtual server with a single web service call.

**Amazon SQS** – Amazon Simple Queue Service is a highly scalable, reliable, hosted queue for storing messages as they travel between computers. It can be used to move data between distributed components of an application that perform different tasks, without losing messages or requiring the components to be always available.

As each submission is made, the video file is stored in Amazon S3 and a message is written in SQS Queue with relevant information so that proper processing of the job can be done. An EC2 instance is created to process the new submission using the information contained in the SQS queue. The message contains relevant information for an EC2 instance process a new job, consisting of transcoding the user's video to a standard format, bitrate and specific codec MPEG4/h.264/AAC [MPEG4-10 2003]. The output video should be easily reproduced by any video player, e.g., Adobe's Flash video player.

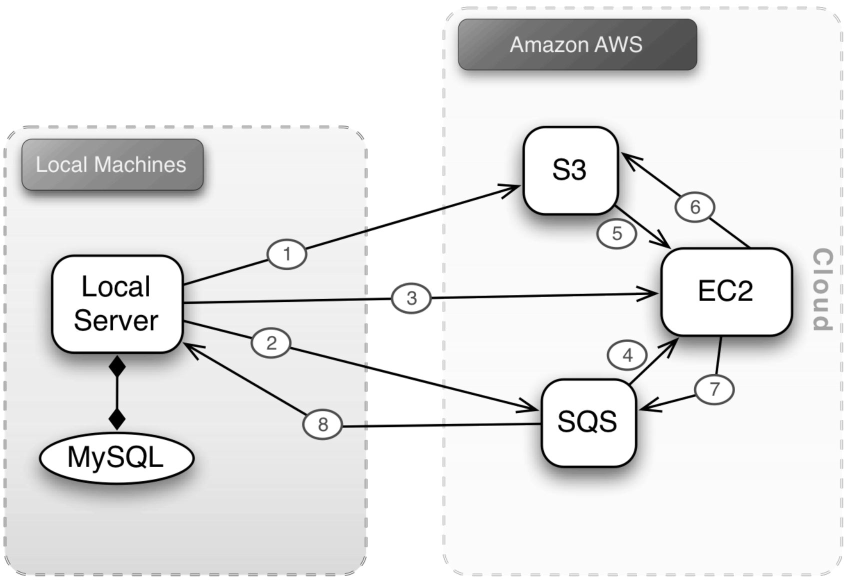


Figure 3.1. Architecture for the public submission system.

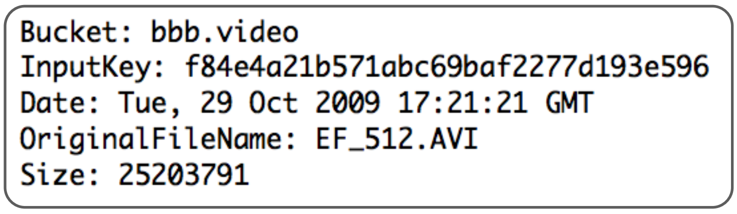
We detail the process in the following basic steps:

1. Video submitted by user is stored in Amazon S3;
2. Local server writes the message in the input queue of SQS detailing the job to be done;

3. Local server creates a new EC2 instance to process the job;
4. EC2 instance reads the message from the input queue;
5. Based on the data of the message the input video is retrieved from S3 and stored locally in the EC2 instance;
6. Video is transcoded by EC2 and the generated output is stored in S3;
7. EC2 instance writes a message in the output queue describing the work performed;
8. The local server from SQS output queue reads confirmation of the work completed.

The local server illustrated in the picture is the web application responsible for receiving the user-generated content.

Messages use the basic structure format used by mail messages and HTTP headers defined in RFC-822 [CROCKER 1982]. Input messages are as follows:

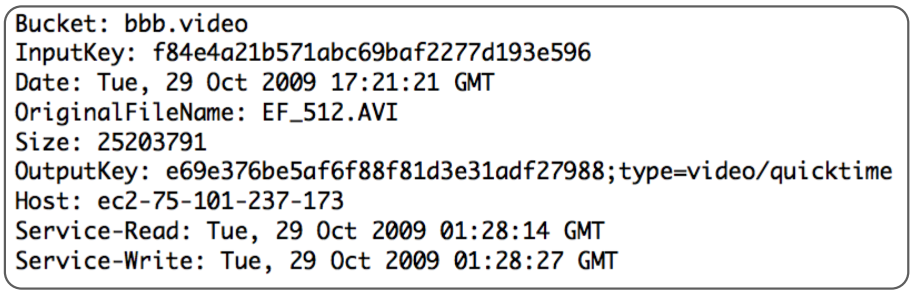


```
Bucket: bbb.video
InputKey: f84e4a21b571abc69baf2277d193e596
Date: Tue, 29 Oct 2009 17:21:21 GMT
OriginalFileName: EF_512.AVI
Size: 25203791
```

Figure 3.2. Example of data in the input message used by the application.

Where *Bucket* and *InputKey* are the identifiers of the file in the S3 infrastructure, and *OriginalFileName* is the source filename. Web services were implemented using the python Boto library [GARNAAT 2006].

The output message is defined as:



```
Bucket: bbb.video
InputKey: f84e4a21b571abc69baf2277d193e596
Date: Tue, 29 Oct 2009 17:21:21 GMT
OriginalFileName: EF_512.AVI
Size: 25203791
OutputKey: e69e376be5af6f88f81d3e31adf27988;type=video/quicktime
Host: ec2-75-101-237-173
Service-Read: Tue, 29 Oct 2009 01:28:14 GMT
Service-Write: Tue, 29 Oct 2009 01:28:27 GMT
```

Figure 3.3. Example of data in the output message used by the application.

Where we also define the hostname of the EC2 instance that processed the job and the timestamps when the job was received and when it finished.

The EC2 instance that is launched uses a specific Amazon Machine Image (AMI) created with all the dependencies necessary to process the video. That includes an updated version of the Linux kernel, *git* to retrieve the latest source code available for this framework, Python and the FFmpeg software, which does the actual video transcoding.

Once the AMI image is instantiated it reads a configuration file that keeps parameters as:

- Command line and arguments – in this case the FFmpeg command;
- Maximum processing time before marking the job as dead;
- Input queue name to read SQS messages;
- Output queue name to store SQS messages;
- Maximum number of retries in case of error;
- Notification e-mail (for debugging purposes);
- Python class to be invoked for the processing.

Due to the generalization of the configuration file, the framework can be setup to a variety of other purposes not restricted to video transcoding.

## 3.4 Underlying Implementation Techniques and Used Technologies

In the complete system there are three different sub-systems:

- Web application for receiving video files from users in the Internet;
- Back-end system to manage the cloud infrastructure, creating EC2 instances, writing/reading in SQS and storing files in S3;
- Video transcoding application of the received content – run in the cloud instance;

The web application was developed using PHP and is the only system with an interface to the end user – the website itself.

The back-end system was written in Python using Boto [GARNAAT 2006] to consume Amazon's web services to manage the cloud infrastructure. The use of Amazon's cloud platform allowed the architecture to be scalable and elastic to fulfill the high peek demand as needed.

Finally, for the transcoding of the video we used FFmpeg, which is a complete solution to convert audio and video [TOMAR 2006]. FFmpeg's libavcodec provides support to a great number of different video formats and codecs.

## 3.5 Description of Presentation

To begin the submission process the user needs to create an account in the reality show's [BBB 10] website.

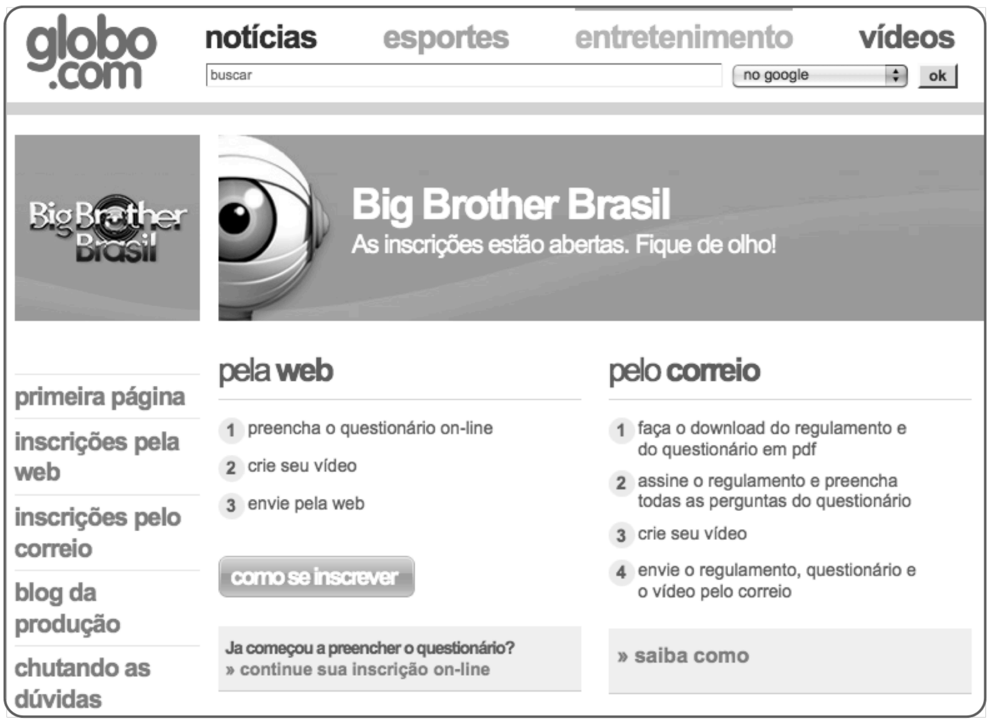
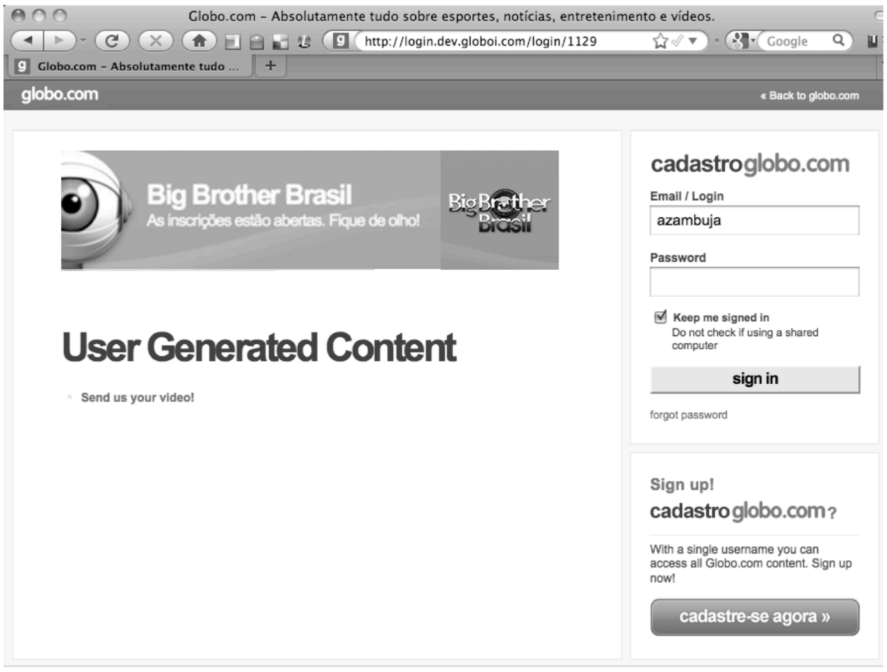

globo.com
notícias
esportes
entretenimento
vídeos
buscar
no google
ok
Big Brother Brasil
Big Brother Brasil
As inscrições estão abertas. Fique de olho!
primeira página
inscrições pela web
inscrições pelo correio
blog da produção
chutando as dúvidas
pela web
1 preencha o questionário on-line
2 crie seu vídeo
3 envie pela web
como se inscrever
Ja começou a preencher o questionário?
» continue sua inscrição on-line
pelo correio
1 faça o download do regulamento e do questionário em pdf
2 assine o regulamento e preencha todas as perguntas do questionário
3 crie seu vídeo
4 envie o regulamento, questionário e o vídeo pelo correio
» saiba como

Figure 3.4. Big Brother Brasil's official web page.

The user can either use an existing account or create a new one. The user account is a requirement so that we can guarantee that the end user is not violating the rights of the content – the user must accept an agreement claiming he is the owner of the content.


Globo.com – Absolutamente tudo sobre esportes, notícias, entretenimento e vídeos.

http://login.dev.globoi.com/login/1129

globo.com

« Back to globo.com

Big Brother Brasil
As inscrições estão abertas. Fique de olho!

User Generated Content

Send us your video!

cadastroglobo.com

Email / Login: azambuja

Password

Keep me signed in
Do not check if using a shared computer

sign in

forgot password

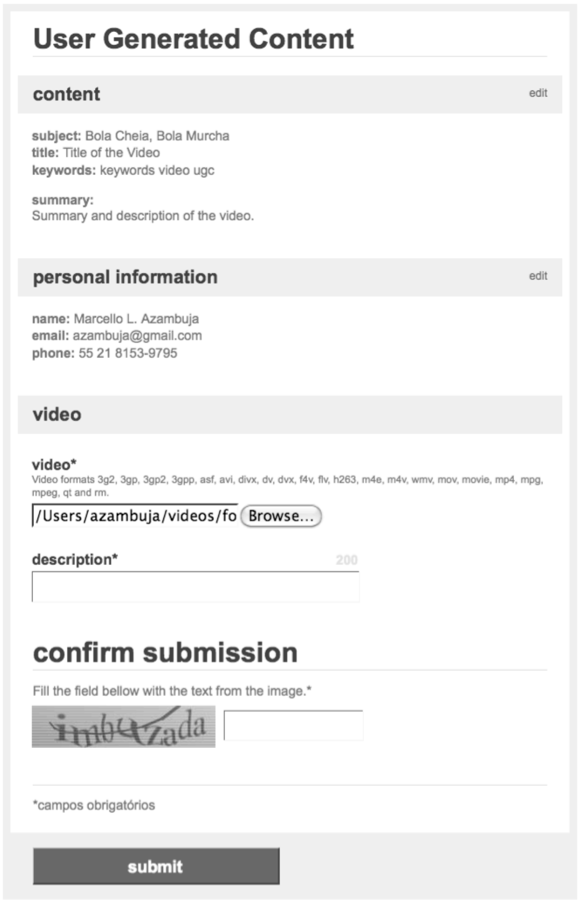
Sign up!
cadastroglobo.com?

With a single username you can access all Globo.com content. Sign up now!

cadastre-se agora »

Figure 3.5. Sign-in and sign-up web page.

Some meta-data information needs to be filled and the video chosen from his local file system.


User Generated Content

content

subject: Bola Cheia, Bola Murcha
title: Title of the Video
keywords: keywords video ugc

summary:
Summary and description of the video.

personal information

name: Marcello L. Azambuja
email: azambuja@gmail.com
phone: 55 21 8153-9795

video

video*
Video formats 3g2, 3gp, 3gp2, 3gpp, asf, avi, divx, dv, dvx, f4v, flv, h263, m4e, m4v, wmv, mov, movie, mp4, mpg, mpeg, qt and rm.

/Users/azambuja/videos/fo Browse...

description* 200

confirm submission

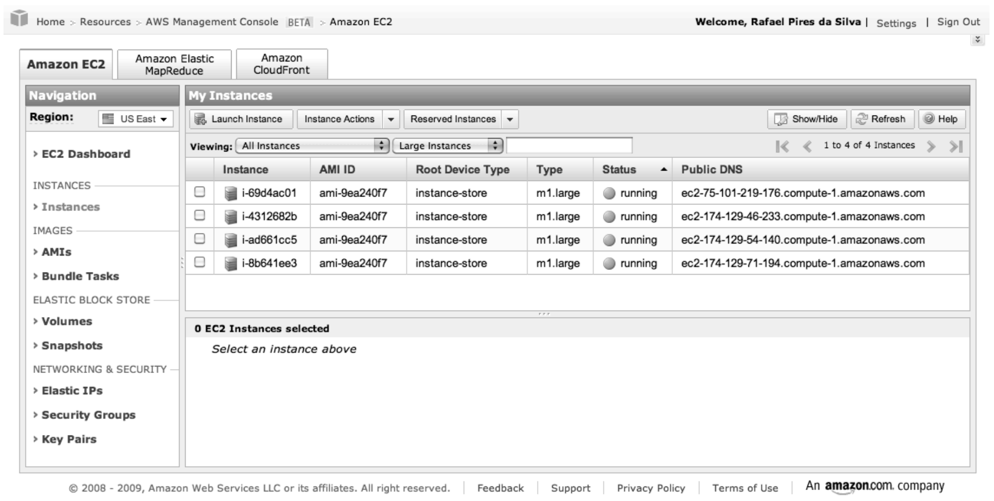
Fill the field bellow with the text from the image.*

*campos obrigatórios

submit

Figure 3.6. Content submission form web page.

Once the video is received by the system, the Cloud is taken into place. Figure 3.7 shows the instances created in Amazon EC2 as the videos were being received.



| Instance | AMI ID | Root Device Type | Type | Status | Public DNS |
|---|---|---|---|---|---|
| i-69d4ac01 | ami-9ea240f7 | instance-store | m1.large | running | ec2-75-101-219-176.compute-1.amazonaws.com |
| i-4312682b | ami-9ea240f7 | instance-store | m1.large | running | ec2-174-129-46-233.compute-1.amazonaws.com |
| i-ad661cc5 | ami-9ea240f7 | instance-store | m1.large | running | ec2-174-129-54-140.compute-1.amazonaws.com |
| i-8b641ee3 | ami-9ea240f7 | instance-store | m1.large | running | ec2-174-129-71-194.compute-1.amazonaws.com |

0 EC2 Instances selected

*Select an instance above*

© 2008 - 2009, Amazon Web Services LLC or its affiliates. All right reserved. | Feedback | Support | Privacy Policy | Terms of Use | An amazon.com. company

Figure 3.7. Amazon AWS management console screen.

Once the transcoding process is completed the instances are shutdown to avoid wasting computing resources from the EC2.

If we were to calculate how much money would be spent to process 100,000 videos with an average size of 15MB, using the small EC2 instance we can process a video in 50% real time, that would require 834 hours of CPU running.

| | | | |
|---|---|---|---|
| Storage | 1.5 TB | U$0.15 / GB | U$ 225.00 |
| Transfer | 1.5 GB | U$0.14 / GB | U$ 210.00 |
| Messages | 200,000 | U$0.01 / 10,000 requests | U$ 0.20 |
| Computer Resources | 834 hours | U$0.085 / hour | U$ 70.90 |
| **Total** | | | **U$ 506.10** |

Table 3.1. Cost analysis of transcoding solution in the Cloud for 100,000 videos.

A total of U$506.10 for transcoding and storing 100,000 videos – that's not even a penny for each video.

Adding to that the fact that no up-front investment and deployment of infrastructure was needed, neither a precise estimation on the expected load of the system we can conclude that Cloud Computing is an excellent solution in this specific scenario. It is important to remark that economical viability of the proposed architecture is such that enables it to quickly deploy at great reduction of the TCO, typical of Cloud Computing implementation [WALKER 2009].