DISSERTATION PAPER

Author: Kedves Szilard

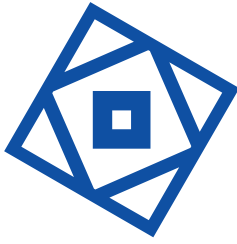**Supervisor:** Associate professor Sasu Lucian Mircea, Ph.D.

Brașov

July 2019

Transilvania University of Brasov

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

DISSERTATION PAPER

Cloud framework for data streaming

Absolvent: Kedves Szilard
Coordonator: Associate professor Sasu Lucian Mircea, Ph.D.

Brașov
July 2019

Contents

Chapter 1

Introduction

1.1   Problem statement

In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions. Hence it is not a surprise that applications that tackle this need are in great demand. These applications use graphical representation of information and data by using visual elements like charts and graphs.

1.2   Motivation

The idea started when I first came into contact with a major cloud service provider: Amazon Web Services also known as AWS.

AWS is a subsidiary of Amazon that offers reliable, scalable, and inexpensive cloud computing services. It provides on-demand cloud computing platforms to individuals, companies and governments, on a metered pay-as-you-go basis. In fact, these cloud computing web services provide a set of primitive, abstract technical infrastructure and distributed computing building blocks and tools. One of these services is Amazon Elastic Compute Cloud, which allows users to have at their disposal a virtual cluster of computers, available all the time, through the Internet. AWS's virtual machines emulate most of the attributes of a real computer including hardware (CPU(s) and GPU(s) for processing, local/RAM memory, hard-disk/SSD storage); a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, etc. [1]

At the beginning its API seemed hard to comprehend, but with time I came to realize how easy it was in fact. Hence my determination to work with as many of its services as possible. At first I started with simple services like S3[TODO legend

for S3], but then I got to understand and use services like ECS[TODO legend for ECS]. The more services I used the more Stream Anlaysis began to expand.

1.3 Thesis structure

Chapter 2

Requirements

In this chapter the functional and nonfunctional requirements for the application are presented.

2.1 Functional requirements

2.1.1 Application overview

Stream Analysis is a web application that provides a hosting environment and defines a workflow for streaming applications, leveraged by cloud.

The hosting environment is responsible for user related sensitive information storage, while it paves the ground for an infrastructure that can handle incoming data. Once information pours in, a dashboard is shown at the user's disposal. This dashboard is updated each time new data arrives, which basically makes it a tool for analytical purposes. The user can visually check if a certain threshold was reached, exceeded, etc.

The workflow is predefined which makes it organized and coherent. Data is fetched or produced and packed by a custom application provided by an external user and funneled in through well defined channels. The stream of data arrives in a secure place where it waits to be queried. This place is located in the cloud so even if one can create the data on-premise, one still needs constant access to the internet. Based on the user's choice the packed information is then retrieved in a real-time fashion or stored. Both ways the data is then accessed and charted. 2D plots are shown with the values at the Y axis, where the X axis holds the timestamps.

It offers two services:

- Push data into cloud - users have to create their own packed software application that streams any type of data into a broker. The broker supports

5 types of protocols ([TODO]more information in chapter ...). Data is organized into topics and queues:

- – Topics - are used to handle real time messages
- – Queues - are used for historical data. Messages are stored and later retrieved

Once the image is ready, the user can proceed with the step by step image upload and use it to create containers.

1. Define topics and queues used in the container
2. Create a repository for the image
3. Push image to repository
4. Add configuration for the image
5. Run image immediately or create a scheduling rule to run the image multiple times over time

Once containers are being created the web application subscribes to the topics defined by the user at step 1. The same happens with queues but in this case the stream of data is dequeued into files so that they can later be retrieved. Stream Analysis also offers the user a dashboard to keep track of his created containers and scheduling rules.

- Visualize data from the cloud - since data is split into topics and queues, the web application takes advantage of this and shows real time plots for topic streams and large scale based plots for queue data.

2.1.2 Access

The Stream Analysis web application is accessible from all around the world as long as the user has internet connection, however it is not open-source and its usage is not free.

The application is targeting two types of users:

- Advanced software engineers - that push data into the application. They need to register with an account and they are charged based on time.

- Normal users - that can read and interpret that data from charts in a dashboard. They also need to register with an account and they are charged based on the network traffic they generate during data visualization.

2.2 Nonfunctional requirements

2.2.1 Diversity

One of its main advantages consists in the ability to feed on any type of data. This asset makes it valuable in any kind of business or domain. For example in the field of meteorology users can plot the weather related data. Having Stream Analysis at their disposal they can track real-time the wind changes. In the field of economics, users can map economical growth of a country next to another one and be able compare, draw conclusions, or forecast.

2.2.2 Performance

The application itself was intended to be very fast and responsive to user interactions. However it has no scaling policies set up so there is a limit of users that can use the web application simultaneously. The limit is around 500 reqs/second.

2.2.3 Security

In my views security is pivotal for any well designed and successful application. Users are guaranteed to not have exposed credentials or any kind of sensitive data. If there is a concern of information leakage, this matter is more deeply discussed in [TO DO Chapter ...].

2.2.4 User's Guide

This paper will softly guide users through its features in further chapters. At this moment it has no stand alone public documentation that users can access.

Chapter 3

Framework architecture

At the highest macro level the framework can be put into perspective as shown in figure 3.1.

Figure 3.1: Highest level of architecture view

The components are:

- Data acquisition users - these are external users that get data from any source and have the responsibility to wrap the fetching into an application and use the framework to upload it

- Packed application - the application made on-premise by the data acquisition user

- Framework - a web application hosted in cloud, used for uploading the packed application, but also to visualize its output

- Data visualization users - the beneficiaries of the wrapped application, they can visualize information with the help of plots
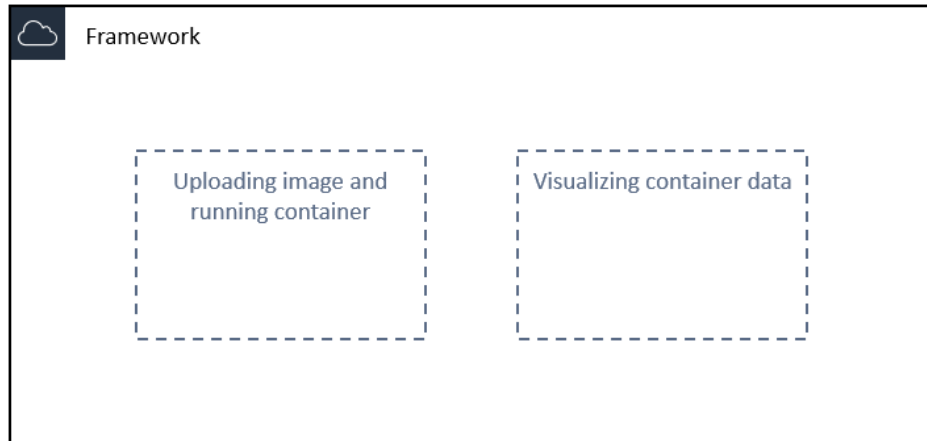
Figure 3.2: Highest level of architecture view

Diving into the web application framework's architecture we have figure 3.2. It has two sides:

- Uploading image and running containers - the data acquisition user after wrapping his fetching data application can use this side of the web application to upload and run it. The steps of achieving this goal is further discussed in chapter 4.

- Visualizing container data - this side of the framework is a tool which can be used to select 2D plots filled with data coming from containers. Selection and usage of these plots is discussed in depth at chapter 4.

Chapter 4

Titlu capitol

Bibliography

[1] *"Amazon Web Services"*. URL: `https://en.wikipedia.org/wiki/Amazon_Web_Services`.