Transilvania
University
of Brasov

FACULTY OF MATHEMATICS
AND COMPUTER SCIENCE

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

Stream analysis web application

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

Brașov
July 2019

Contents

1 Introduction **2**
1.1 Problem statement and Motivation 2

2 Requirements **3**
2.1 Nonfunctional requirements 3
2.1.1 Access 3
2.1.2 Performance 3
2.1.3 Guide 4
2.2 Functional requirements 4
2.2.1 Application overview 4

3 Titlu capitol **5**

4 Titlu capitol **6**

Chapter 1

Introduction

1.1 Problem statement and Motivation

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.

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.

Chapter 2

Requirements

In this chapter the basic user requirements are presented.

2.1 Nonfunctional requirements

2.1.1 Access

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

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 value of the data they visualize.

2.1.2 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 req/second.

2.1.3 Guide

It has no stand alone public documentation that users can access, but this paper will softly guide users through its features in further chapters.

2.2 Functional requirements

2.2.1 Application overview

Stream Analysis is a web application hosted in the cloud which offers two services:

- Push data into cloud - users have to create their own Docker image that streams any type of data into an ActiveMQ 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.

  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 container
  5. Run image
     (a) Immediately
     (b) Create a scheduling rule to run the image multiple times over time

  Once containers are being created the web application subscribes to the topics inserted 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 static time based plots for queue data.

Chapter 3

Titlu capitol

Chapter 4

Titlu capitol

Bibliography

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