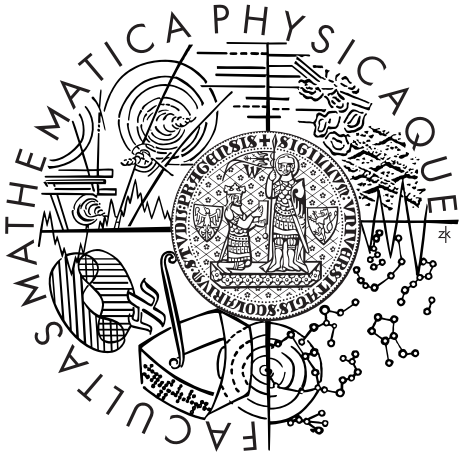
Charles University in Prague
Faculty of Mathematics and Physics

**MASTER THESIS**




Ondřej Klejch

# Development of a cloud platform for automatic speech recognition

Institute of Formal and Applied Linguistics

Supervisor of the master thesis: Mgr. Ing. Filip Jurčíček Ph.D.
Study programme: Informatics
Specialization: Theoretical Computer Science

Prague 2015

Dedication.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date .....  signature of the author

Název práce: Development of a cloud platform for automatic speech recognition

Autor: Ondřej Klejch

Katedra: Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: Mgr. Ing. Filip Jurčíček Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt:

Klíčová slova:

Title: Development of a cloud platform for automatic speech recognition

Author: Ondřej Klejch

Department: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Ing. Filip Jurčíček Ph.D., Institute of Formal and Applied Linguistics

Abstract:

Keywords:

# Contents

# Introduction

The most natural form of human communication is speech. In order to be able to talk with a computer, it is crucial to have a good Automatic Speech Recognition (ASR) system. On one hand, there are several open-source ASR toolkits, however deployment of such toolkits requires substantial knowledge therefore for common software developers it is not easy to use them. On the other hand, there are a few webservices that provide ASR as a service, yet these webservices do not solve all problems - either they are paid, closed-source or they are not customizable. So **the first goal of the thesis is to develop a cloud platform for ASR** that is easy to use both from user's and maintainer's point of view.

Although accuracy of ASR systems is improving, these systems are still far from perfect. One of the reasons is that accuracy of ASR systems relies heavily on the amount of the training data and there is not enough publicly available transcribed speech data. By providing free ASR webservice it is possible to collect vast amount of recordings that can be manually transcribed and used later on for further research. Consequently, **the second goal of the thesis is to create an annotation interface** so that recordings obtained by CloudASR platform can be annotated and given back to the community.

In the following text there will be described development and deployment of CloudASR platform and of its annotation interface. Chapter 1 introduces ... In Chapter 2 architecture of CloudASR is described. Annotation interface and theory related to obtaining of human transcriptions is presented in Chapter 3. Finally, Chapter 5 concludes this thesis. User manual and programmer manual can be found in the Attachments.

# 1. Automatic Speech Recognition

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

## 1.1 Acoustic Models

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

## 1.2 Language Models

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in

of the original language. There is no need for special content, but the length of words should match the language.

## 1.3 Decoding

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 1.3.1 Batch Decoding

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 1.3.2 Online Decoding

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

## 1.4 Transcriptions

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

## 1.5 Transcriptions via Crowdsourcing

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 1.5.1 Amazon Mechanical Turk

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 1.5.2 CrowdFlower

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

## 1.6 Open-Source ASR tools

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 1.6.1 HTK

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information.

Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 1.6.2 RWTH

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 1.6.3 Sphinx

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 1.6.4 Kaldi

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 1.6.5 PyKaldi

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet

and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

## 1.7 ASR cloud services

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 1.7.1 Google Speech API

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 1.7.2 Nuance

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 1.7.3 Tom Robinson

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 1.7.4 Wit.ai

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

# 2. Used Technologies

In this chapter technologies that were used during development will be described. Also the motivation for the usage of these technologies will be explained.

## 2.1 Platform

In the following section technologies that were used to build a cloud platform will be described. These technologies made possible to build a scalable solution with an easy deployment.

Traditionally, a deployment of a such a complex system as CloudASR is consists of several steps during which necessary dependencies are installed, the application environment is set up and finally the application is started. But this approach makes the maintenance of these systems difficult, because the deployment is time consuming, error-prone and it is not replicable. The ultimate goal for the CloudASR deployment was the exact opposite: fast and replicable deployment.

The most important tool used during development was **Docker** [6] – a portable, lightweight application runtime and packaging tool. It allows to specify dependencies and environmental variables for a process and it allows to build a image from this specification called Dockerfile, see Figure 2.1 for an example. Once this image is built it can be used on any machine with Docker installed, which makes the deployment fast and replicable, because it is not neccessary to install all dependencies on every machine. Additionally, the usage of Docker images removes bugs caused by different versions of libraries used in development and production environmnent because developers use the same images in both environments.

```
FROM ubuntu
MAINTAINER Ondrej Klejch

RUN sudo apt-get update && sudo apt-get install python
ADD . /opt/app
WORKDIR /opt/app

CMD python run.py
```

**Figure 2.1:** *An example of Dockerfile that creates an image from the base ubuntu image, installs python, copies all files in the Dockerfile folder and sets command python run.py to be run, when the docker image is started.*

When running an application in a cloud it is neccessary to monitor all servers and handle failovers. But with the increasing number of servers the maintenance costs grows rapidly. Therefore it is not possible to do that manually. The tool that allows CloudASR to run on many servers is **Mesos** [5]. It lets users program against set of machines like it is a single machine, which means that it is possible to run and scale an application on a set of servers in a similar way as on a single machine. Mesos takes care about scheduling and high availability of the platform.

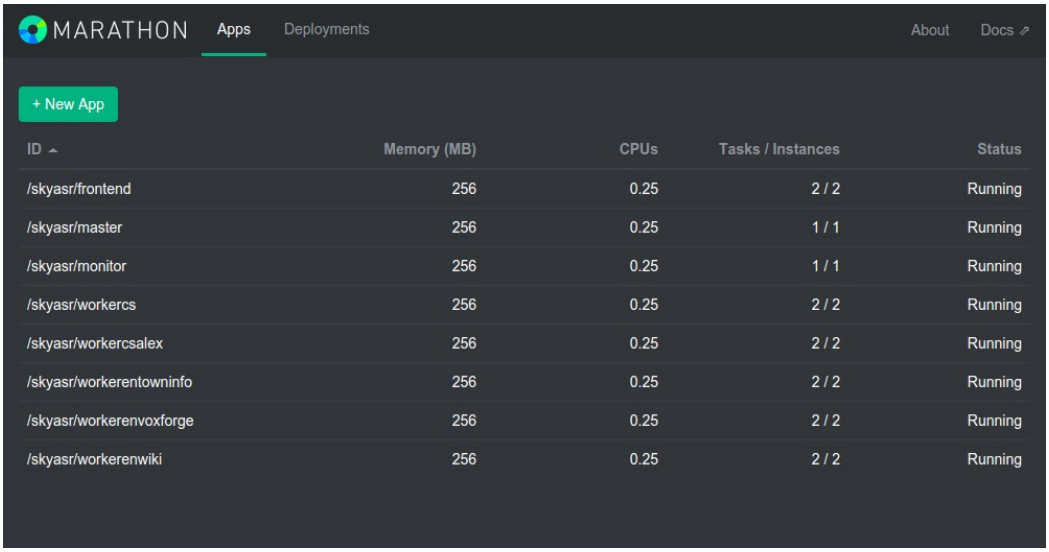



**Figure 2.2:** *A screenshot of a Marathon web interface with a running CloudASR platform.*

Thus, whenever some part of the CloudASR crashes Mesos will try to fix that. Finally, Mesos supports Docker so the images that are used in development can be also used on a Mesos cluster.

**Marathon**[1] is a framework built on top of Mesos whose main responsibility is to launch long running applications. It is an entrypoint for running and scaling the applications running on a Mesos cluster. It has a web user interface (see Figure 2.2) and REST API, through which applications can be started, scaled or stopped easily.

Since the traffic of CloudASR platform can be very large, it is not possible to process all HTTP requests by one application server. Therefore, there must be a load-balancer to distribute workload between application servers. CloudASR platform uses **HAProxy**[2] load-balancer, but any other load-balancers can also used with appropriate setup.

## 2.2 Continuos Integration & Delivery

During development of CloudASR several practises were obeyed, namely Continuous Integration and Continuous Delivery. In order to be able to do that a platform which consisted of **Jenkins-CI**[3] and **Docker Registry**[4] had to be deployed.

The most important tool for Continous Integration & Delivery of CloudASR is Jenkins-CI. Its task is to watch CloudASR git repository and whenever a new code is pushed into this repository it schedules a new build of the platform. During this build the most recent code is pulled from the repository and then the docker images are built. After that tests are run to check that the new code did

[1] `https://mesosphere.github.io/marathon/`
[2] `http://www.haproxy.org/`
[3] `https://jenkins-ci.org/`
[4] `https://github.com/docker/docker-registry`

not break anything. Finally, successfully built images are tagged with current build number and pushed to the Docker Registry.

Docker Registry is a repository of Docker images. Even though, there are several Docker Registry providers[5], which are free for open-source projects, CloudASR uses its own free Docker Registry, in order to be also able to use proprietary software that cannot be shared with public.

## 2.3 Backend

The main programming language used for development is **Python**[6]. Web and REST API are built on top of **Flask**[7] microframework and they use **Gunicorn**[8] for production deployment.

CloudASR architecture consists of several nodes which need to communicate between each other. For this communication ClousASR uses **ZeroMQ**[9], because of its simple design, high performance and support for every modern language. With ZeroMQ it is possible to create many messaging patterns, but CloudASR uses only two: request-reply and push-pull. These patterns are described in detail on Figure 2.3.

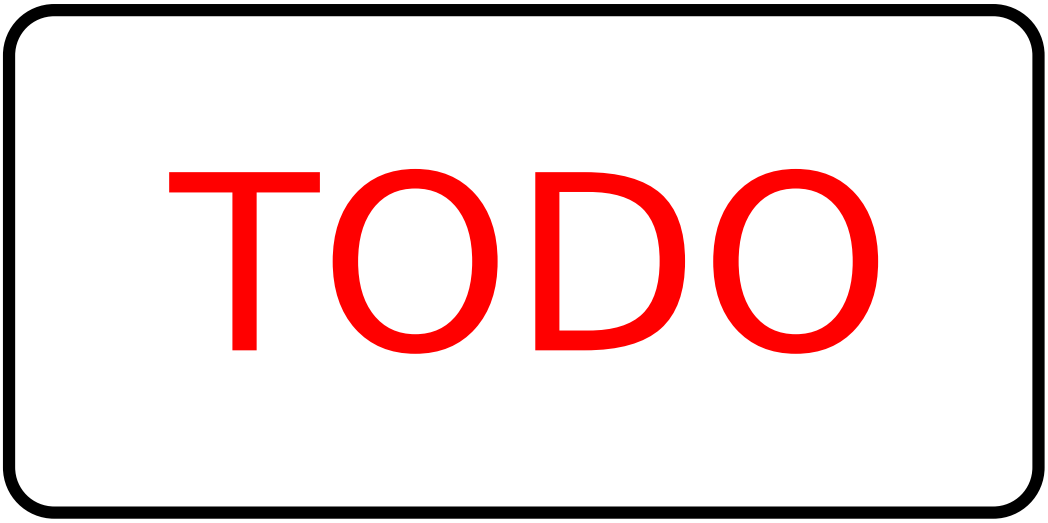



**Figure 2.3:** *Description of used ZeroMQ patterns.*

In order to be able to send complex messages via ZeroMQ sockets, messages have to be serialized. CloudASR uses **Google Protocol Buffers**[10], because they have support in many languages, allow specification of various message types (See Figure 2.4 for example) and serialize messages in very compact way (See Table 2.1 for a comparison of different serializations).

As an ASR toolkit CloudASR uses Python wrapper for the **Kaldi speech recognition toolkit** [8] called **Pykaldi** [7]. Because CloudASR should be able

[5] `https://hub.docker.com/`, `https://quay.io/`
[6] `https://www.python.org/`
[7] `http://flask.pocoo.org/`
[8] `http://gunicorn.org/`
[9] `http://zeromq.org/`
[10] `https://developers.google.com/protocol-buffers/`

```
message HeartbeatMessage {
  required string address = 1;
  required string model = 2;
  required Status status = 3;

  enum Status {
    STARTED = 0;
    WAITING = 1;
    WORKING = 2;
    FINISHED = 3;
  };
}
```

**Figure 2.4:** *An example of Google Protocol Buffer message specification with three fields. Fields address and model are just strings and the status is an enum with four possible values.*

| | | |
|---|---|---|
| **raw file size** | 56146 | |
| **bytes_protobuf** | 56118 | 0.999x |
| **base64** | 74872 | 1.333x |
| **json_array** | 158590 | 2.824x |

**Table 2.1:** *The table shows comparison of different serialization used to serialize a wave file into a message for the CloudASR online mode. As can be seen from the results Google Protocol Buffers achieved the best result.*

to process very long recordings recordings, possibly infinite, with a limited computational resources it is necessary to split the recordings into smaller chunks. For that purpose CloudASR uses voice activity detector implemented in **Theano** [1] to detect silences in a speech.

## 2.4 Frontend

Frontend uses several well-known open-source libraries, namely, **Twitter Bootstrap**[11] for CSS styling of the web, **jQuery**[12] and **Angular.js**[13] for interactive elements on the web.

Modern web browsers supports **WebAudio API**[14], which is a high-level JavaScript API for processing and synthesizing audio in web applications. One of the things that can be done with this API is recording of an audio. Thus, it is possible to create a web demo for the CloudASR online mode. The demo is based on **Recorder.js**[15] library, which can record output of WebAudio API and return it as a PCM chunks.

Next step is to send these chunks to the API. Because the demo demonstrates

[11] `http://getbootstrap.com/2.3.2/`
[12] `https://jquery.com/`
[13] `https://angularjs.org/`
[14] `http://webaudio.github.io/web-audio-api/`
[15] `https://github.com/mattdiamond/Recorderjs`

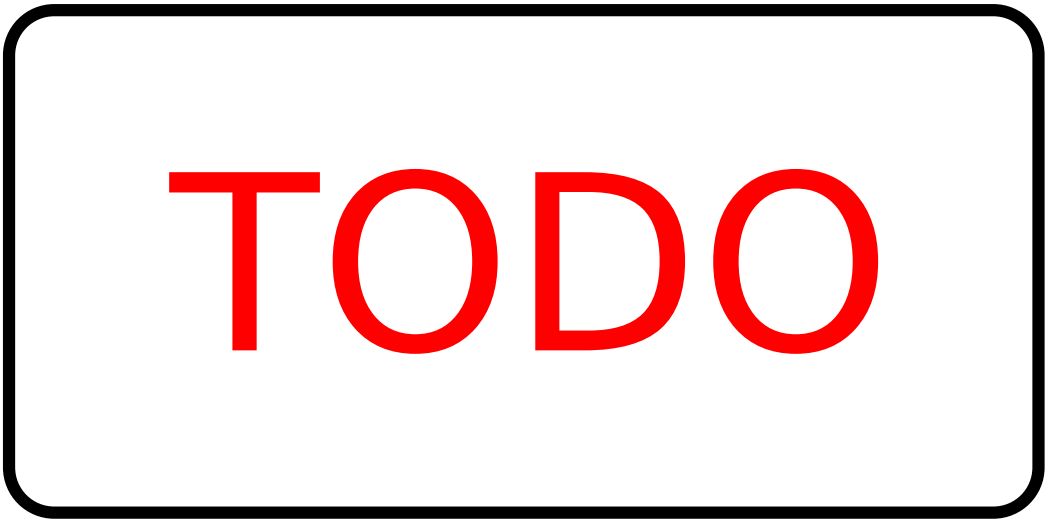



**Figure 2.5:** *CloudASR Web Demo*

the online speech recognition mode, it is not possible to wait for whole recording to be recorded and then send it to the API via HTTP POST request, thus, CloudASR uses **Socket.IO**[16] to send stream of chunks to the API and to receieve stream of results from the API.

[16] `http://socket.io/`

# 3. Solution

In this chapter the CloudASR implementation will be described. The implementation was affected by three key requirements:

- **Scalability** - because the speech recognition is a demanding process in terms of computational resources, it is not possible to handle many parallel requests on one machine. Therefore the CloudASR architecture is designed to be able to scale across many machines.
- **Customizability** - there are already several webservices that provide an API for speech recognition, but they are not easily customizable. Thus, the second requirement was to be able to host any Kaldi model on the CloudASR platform. Moreover, the CloudASR platform is able to run any ASR system, if the users implement a wrapper for that system.
- **Easy deployment** - complex systems as CloudASR have many dependencies and difficult deployment process, which makes their maintenance hard. CloudASR is designed to has as few dependencies as possible and an one command deployment.

## 3.1 Architecture

In order to meet the aforementioned requirements the platform had to be designed from the very beginning to be able to run on many machines. To be able to do that the architecture consits of several nodes that communicate with each other by sending messages over ZeroMQ sockets [4]. Additionally, it is not possible start ASR system when the user sends request, because the ASR systems need some time load decoding graphs in memory and this would add some unnecessary latency. Therefore the platform uses Master-Worker architecture to be able to handle many parallel requests for various languages.

The CloudASR architecture as described in Figure **??** consists of several types of nodes that can run on different machines. These nodes are **Master**, **Worker**, **API**, **Web**, and **Recordings Saver**. In the following section each node will be described in detail.

### 3.1.1 Master

**IDEA:** queue for every language

The main task of the Master is to keep track about running workers and scheduling tasks to them. The Master keeps track about running workers which send heartbeats with information about their state and when the Api asks for a worker address the master can return address of an available worker.

The workers can be in four different states: **started**, **waiting**, **working** and **not responding**. and it sends four different heartbeats: **started**, **waiting**, **working** and **finished**. The worker starts in in the **started** state, after that it moves to the **waiting** state by sending the **waiting** heartbeat. The worker

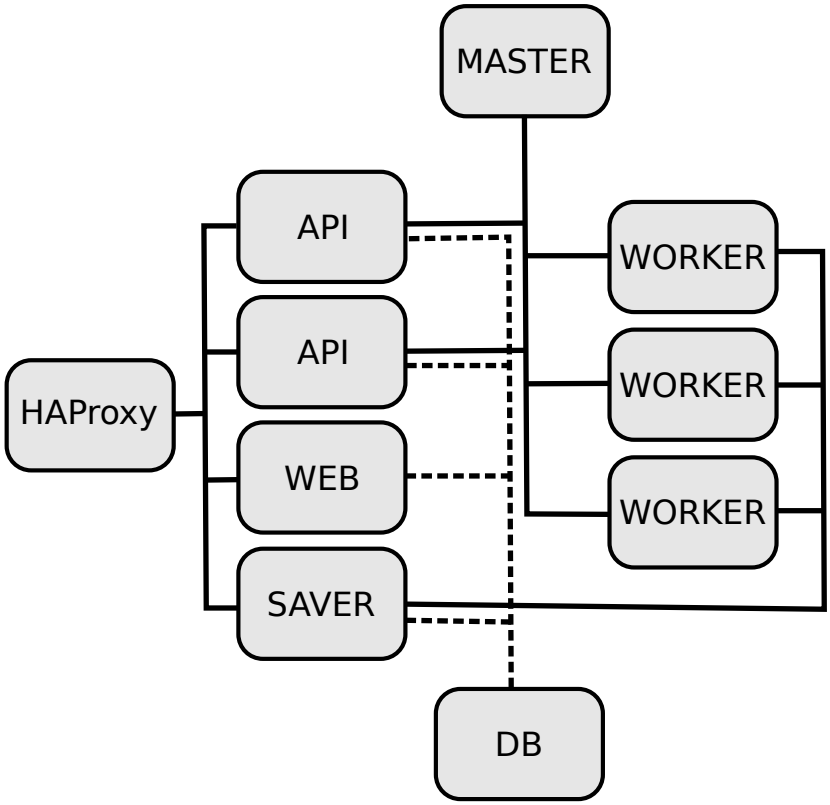



**Figure 3.1:**

**TODO:** Architecture

remains in the waiting state until **the Master assings a tasks** to it, then it moves to the **working** state where it remains as long as it is working. In the working state worker sends working heartbeats periodically, to inform the Master that it is working and it did not fail. At the end of the task the Worker sends **finished** heartbeat and the Master changes the state of the Worker to the **working** state.

Additionally, when a worker crashes during the processing of the task and it gets restarted, it sends started heartbeat again, which informs the Master, that the worker was restarted and it adds it to the queue again.

When a worker does not send any heartbeat for 10 seconds, the master set the worker state to **not responding**. But as soon as the worker sends any heartbeat, the master will set the worker to the appropriate state.

This whole process can be seen as a finite automaton illustrated in Figure **??**.

Unfortunately, the Master is a single point of failure of CloudASR platform. When the Master stops working no speech recognition requests can be processed, because the API containers will not know to which worker they can forward the request. But as soon as the Master starts working the platform should be available again.

### 3.1.2 Worker

**IDEA:** VAD

**IDEA:** Kaldi

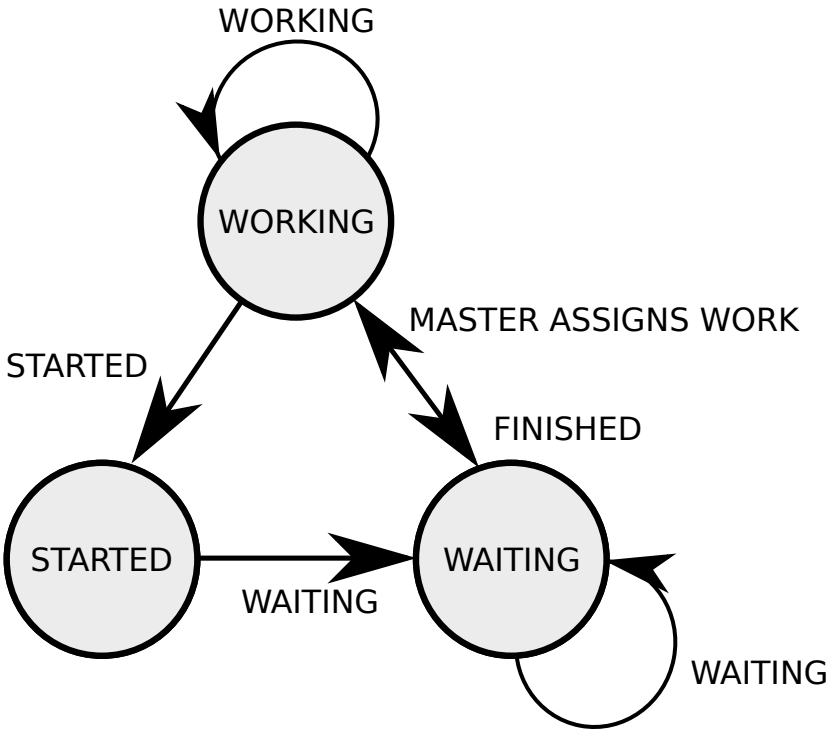



**Figure 3.2:**

**TODO:** Worker state transition diagram

**IDEA:** Heartbeats

Heartbeats that Worker sends to Master are small messages with Worker address, language it processes and its state.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 3.1.3 Deployment of New Kaldi Worker

One of the requirements for the CloudASR was easy customizability, primarily easy customizability of the workers decoding graphs. Therefore, CloudASR supports an easy way for users to add their own Kaldi models. The only thing that the users has to do is to create a worker docker image with their models. This process consists of a few steps. First, users have to create a script `download_models.sh` which will download all necessary files from their server, see Figure 3.3 for example. Second, they have to create a configuration file `config.py` with appropriate configuration for the downloaded models, see Figure 3.4. Finally, they have to copy a Dockerfile (see Figure 3.5for example) for the worker and build the docker image with following command: `docker build -t new_worker .`. After that users can use the new worker in their application in the similar way as they use

other models.

It is important to note that the new worker docker image will be available only on the machine where it was built. If the users want to use this model on multiple machines, they have to push to their docker registry with command `docker push ...` or alternatively they can update Jenkins scripts `build_workers.sh` and `push_workers.sh` to also build their new worker.

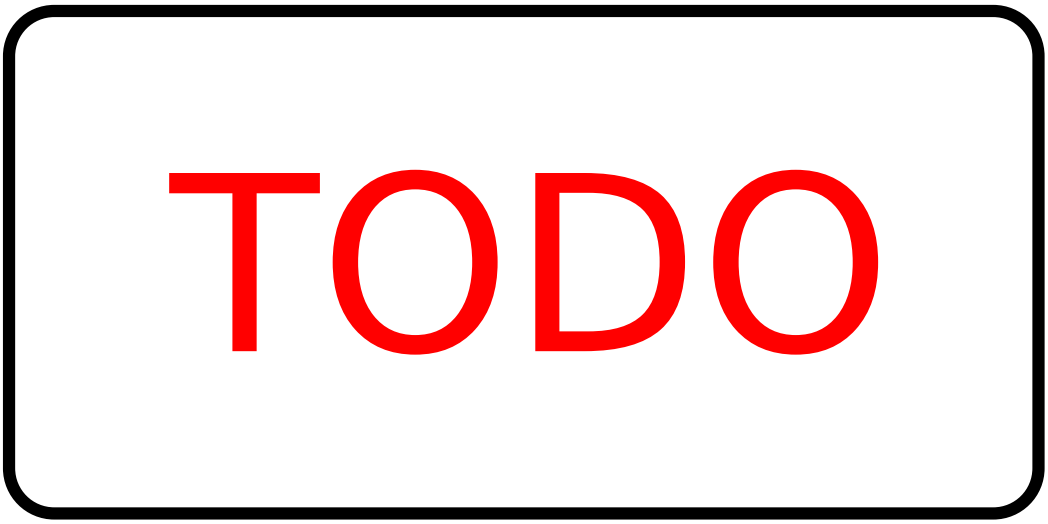



**Figure 3.3:** *Example of download_models.sh script*

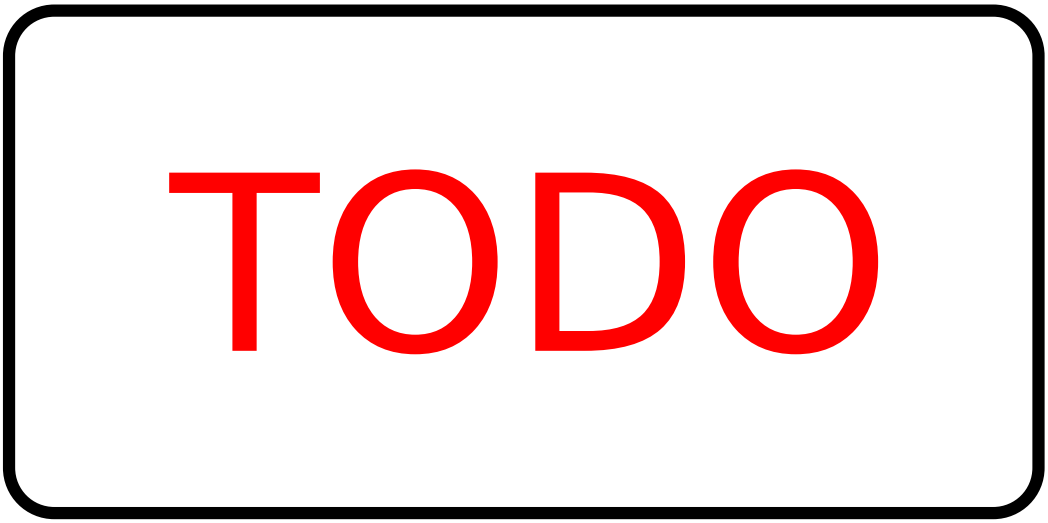



**Figure 3.4:** *Example of config.py script*

### 3.1.4 Deployment of Arbitrary Worker

Even though CloudASR support only Kaldi out of the box, other ASR systems can be used too. Again, the only thing that the users has to to is to create a worker docker image with their ASR system. The only step that differs from the previous process is that the users have to implement and add to the Dockerfile a script with their own `create_asr` method that returns `ASR` class with these methods:

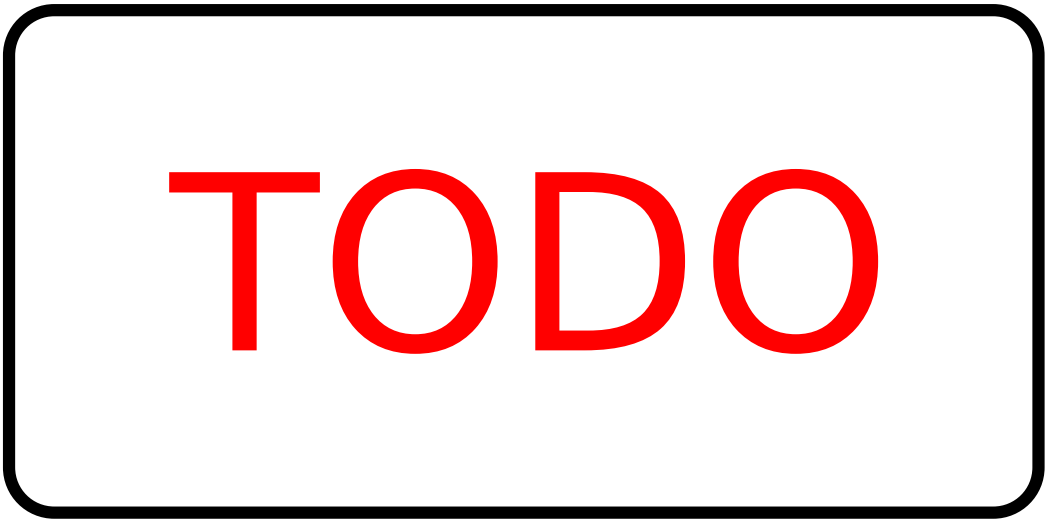



**Figure 3.5:** *Example of worker dockerfile*

- `reset()`

  **TODO:** add description

- `recognize_chunk(pcm)`

  **TODO:** add description

- `get_final_hypothesis`

  **TODO:** add description

The process is illustrated on Figure 3.6 on the DummyASR class, which will be also used for benchmark purposes in the Chapter 4.

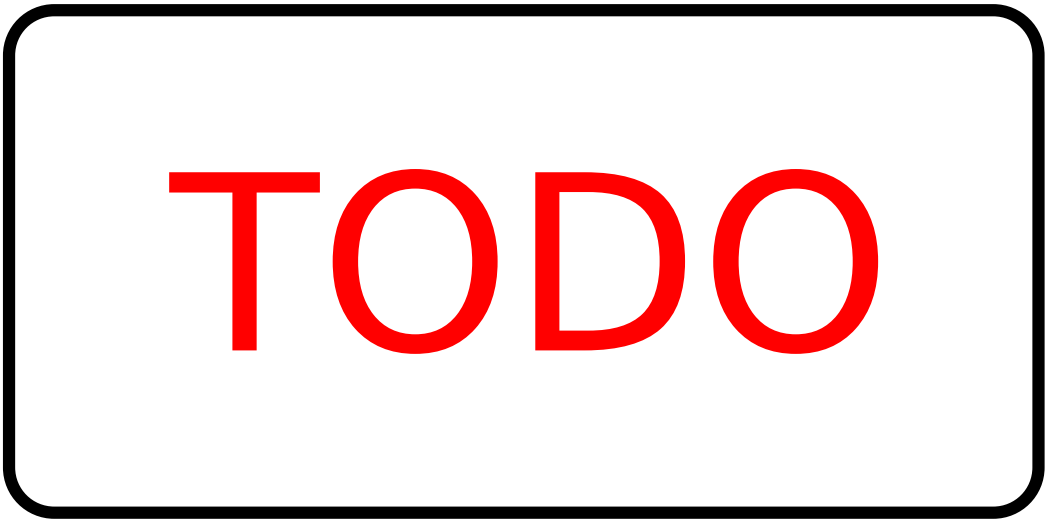



**Figure 3.6:** *Example of DummyASR class*

### 3.1.5 API

**IDEA:** not blocking architecture

The main task of the API container is to forward requests from the clients to the workers. The requests are either in form of HTTP POST for the batch mode or Socket.IO for the online mode. The API is built on top of Flask framework with enabled asynchronous processing which allows single API container to process many parallel requests, because there are no blocking operations in the API container - it just receives requests from clients and forwards them via ZeroMQ to the workers.

### 3.1.6 Web

**IDEA:** Mention Annotation Interface and Webdemo also link to screenshots

**IDEA:** Describe process of annotation

CloudASR platform also has a web interface. The web interface distinguishes between two types of user roles: administrators and users. Administrators can browse all recordings with their transcriptions and manage workers descriptions. Normal users are only allowed to add transcriptions to the recordings.

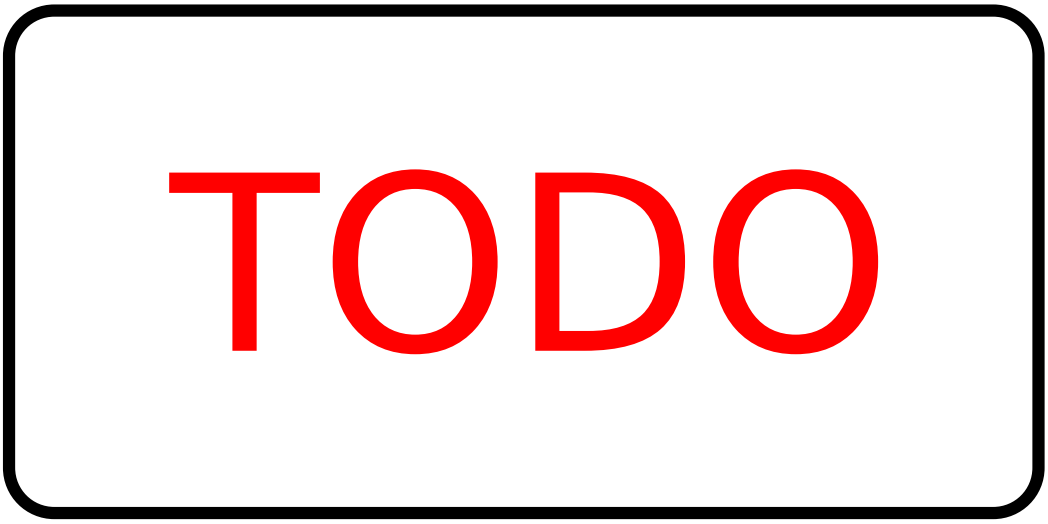



**Figure 3.7:** *Screen of the Annotation Interface*

Finally, the Web hosts a demo (See Figure 3.8) of the CloudASR platform, through which users can try out different workers directly in their web browsers. The demo has two modes, namely, dictation mode that only shows the best transcription of the recording and evaluation mode that also allows users to confirm that the recording is correct.

### 3.1.7 Recordings saver

The main task of the Recordings saver is to save and serve recordings processed by workers. When the worker finishes recognition it sends the recording with its n-best hypotheses to the Recording saver via ZeroMQ socket. The saver saves

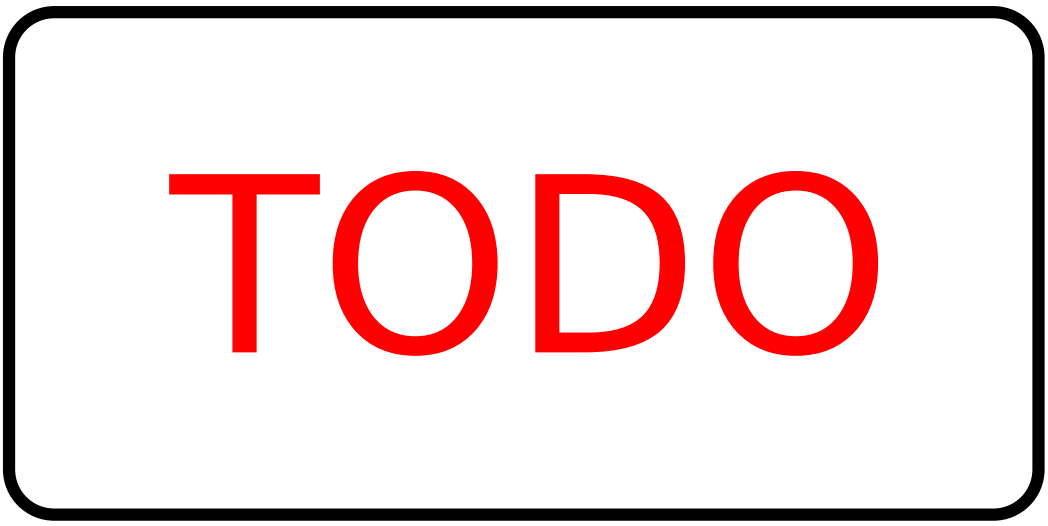



**Figure 3.8:** *Screen of a Web Demo*

the wave file to the filesystem and it save the n-best hypotheses to the database so that they can be used in the future.

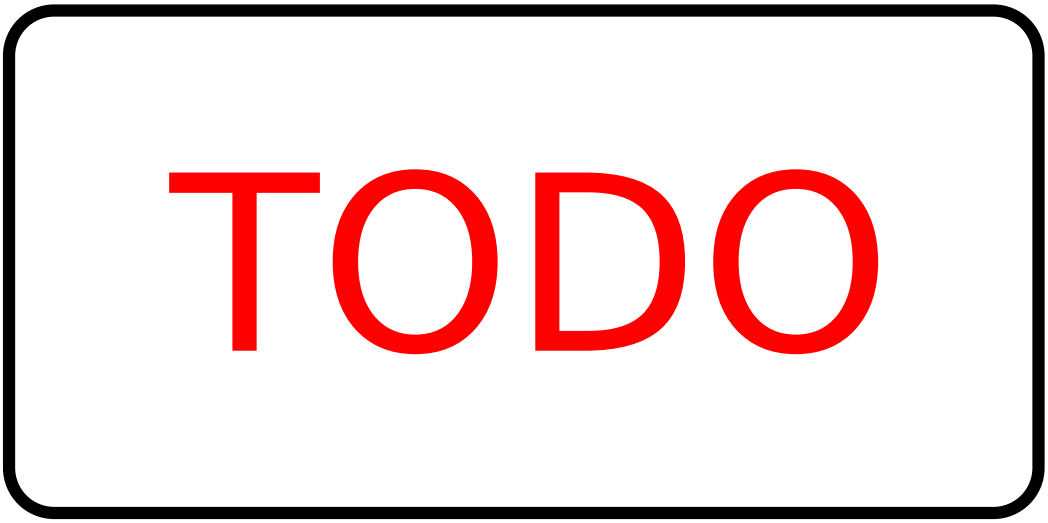



**Figure 3.9:** *Database schema*

## 3.2 Request Workflow

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 3.2.1 Batch Recognition

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

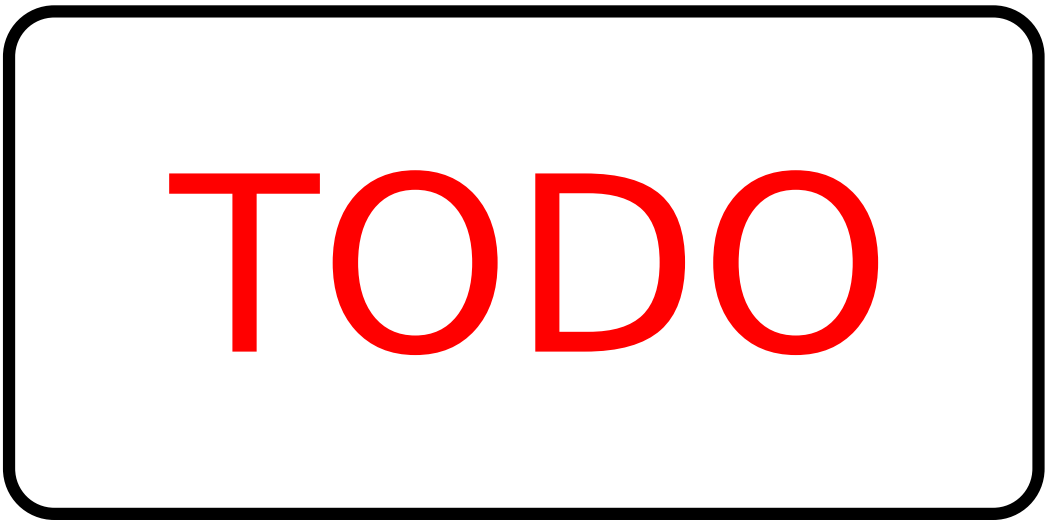



**Figure 3.10:** *Batch Workflow*

### 3.2.2 Online Recognition

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

## 3.3 Deployment

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

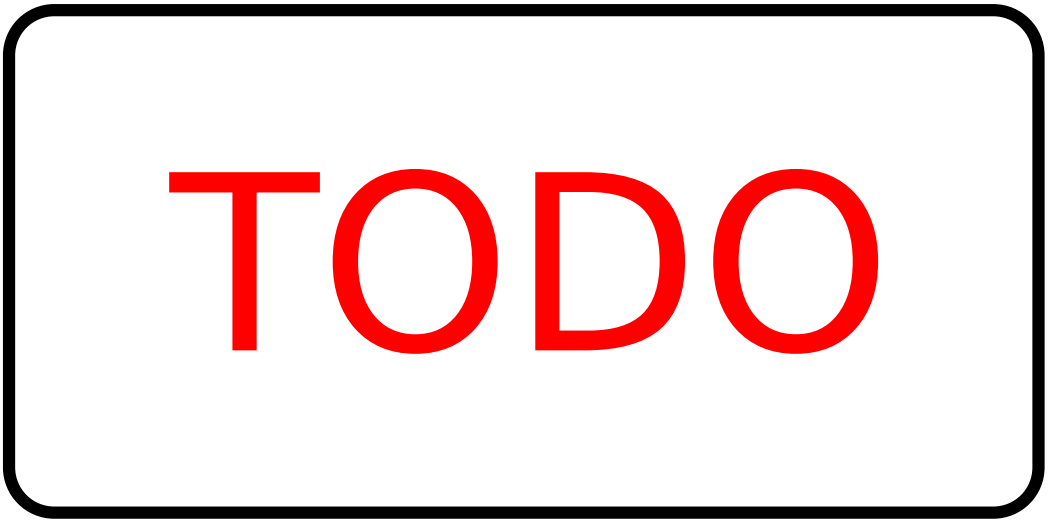



**Figure 3.11:** *Online Workflow*

### 3.3.1 Single-Host Deployment

Single-Host deployment allows users to run CloudASR directly on their machines. The only dependency for running CloudASR is Docker. After that it is possible to use prepared Makefile to run CloudASR by command `make run_locally`. The platform can stopped by command `make stop`.

The Makefile is prepared to run only one worker with English TownInfo model. In order to run different worker users have to edit the Makefile, especially the name of the worker Docker image.

**TODO:** don't use Makefile, use some python script with same interface for Multi-Host Deployment

### 3.3.2 Multi-Host Deployment

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

**TODO:** describe Mesos installation

**TODO:** insert example cloudasr.json configuration for the deployment

### 3.3.3 Scalability

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this

```
def test_worker_sends_heartbeat_after_finishing_task(self):
    messages = [
        {"frontend": self.make_frontend_request("message 1")}
    ]

    self.run_worker(messages)
    self.assertThatHeartbeatsWereSent(["STARTED", "FINISHED"])
```

**Figure 3.12:** *An example of unit test structure.*

**TODO:** add better caption

gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 3.3.4 Implementation process

Each node was implemented with two main design patterns in mind, namely Dependency Injection [2] and Factory Method [3]. Usage of these patterns together with message oriented architecture made possible to test the whole platform easily, because it enabled to pass test doubles into the nodes and then send fake messages needed to test a correct behaviour. As a result, a typical unit test structure looks like a test in Figure **??**.

In addition to unit tests there are also integration tests, which test the factory methods that create production ready nodes, and end-to-end tests, which test that both batch and online recognition mode requests are handled correctly. This test suite ensures that developers did not break anything and it also gives them confidence to change the code without fear.

### 3.3.5 Contiunous Integration & Countinuous Delivery

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

## 3.4 Discussion

**IDEA:** Compare throughput of Queue vs Master-Slave, discuss options

**IDEA:** different architectures solving Harddisk bottleneck, Network bottleneck

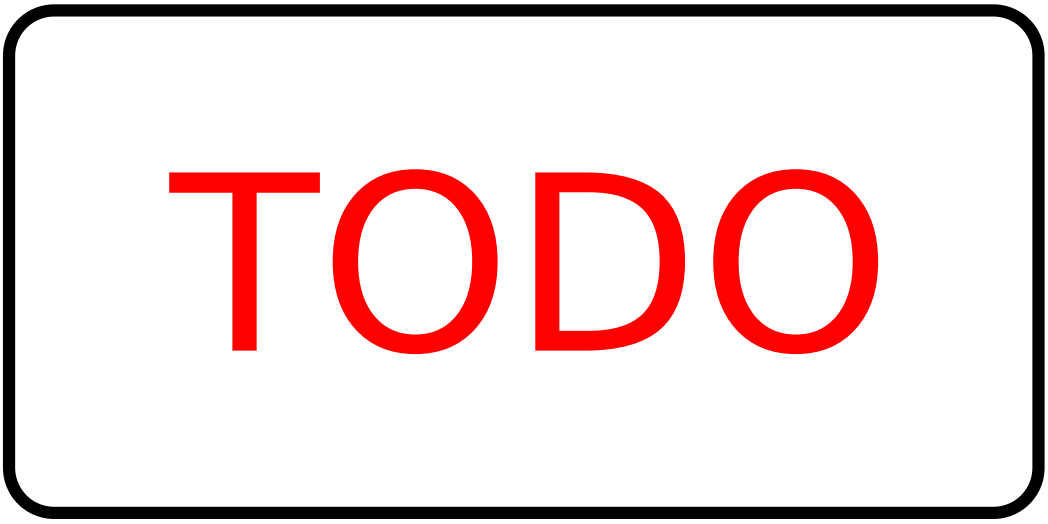



**Figure 3.13:** *Queue architecture*

# 4. Evaluation

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

## 4.1 CloudASR Platform Benchmarks

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

## 4.2 Batch Recognition Benchmark

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in

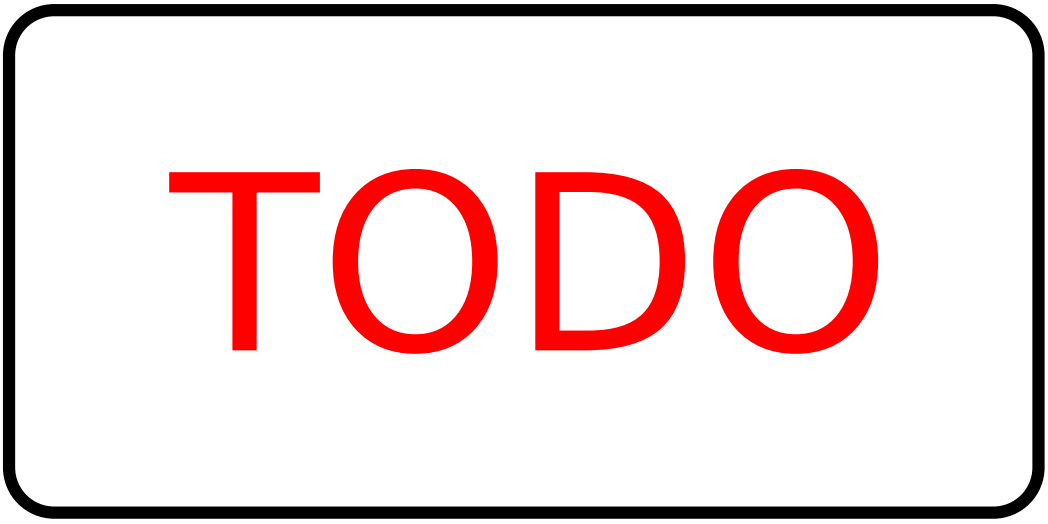



**Figure 4.1:** *fig:batch-benchmark*

of the original language. There is no need for special content, but the length of words should match the language.

## 4.3 Online Recognition Benchmark

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some


**Figure 4.2:** *fig:online-benchmark*

text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there

no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

# Conclusion

Goals of this thesis were to develop a cloud platform for ASR, CloudASR, and an annotation interface for annotating speech data. These goals were successfully accomplished and in several aspects even surpassed - in addition to original requirement to create batch recognition mode, we also implemented online recognition mode. In the following sections we summarize our achievements in detail and at the end we propose ideas for future work.

## Cloud platform for ASR

The first goal of this thesis was to develop a cloud platform for ASR, CloudASR, that would provide batch API for speech recognition of wave files. The platform uses Master/Worker architecture. Consequently, it is able to run both on single-machine and multi-machine setup. The platform allows us to run workers for various language models and to scale workers according to our needs. To be able to run CloudASR on several machines we chose Mesos/Marathon as an underlying technology The current implementation of the API supports two modes of speech recognition: batch and online.

Firstly, batch mode allows users to send a file with a recording to the server and then it sends transcribed text back as a json. API of this mode is similiar to Google Speech API which allows users to switch from Google Speech API to CloudASR easily.

Secondly, users can transcribe speech recordings in real-time via online mode. We have also created Python and JavaScript libraries for using our API. JavaScript library achieves similiar latency as WebkitSpeechRecognition in Google Chrome

**TODO:** add benchmark

.

Finally, we wanted CloudASR to be easily deployable. Because of that, we used Docker for creating and running application containers. As a result only dependency that users have to install is Docker for single-node setup and Mesos Cluster for multi-node setup. Moreover, installation scripts for these dependencies are included within the distribution together with deployment scripts, that can be used for CloudASR instances management.

## Annotation interface

The second goal of this thesis was to create an annotation interface for annotating speech data. First responsibility of the annotation interface is to collect and store obtained recordings.

The second responsibility is to allow users to rate transcriptions of the recordings (Is the transcription correct? yes/no) or to subsequently add their own transcriptions. The annotation interface implements algorithm to choose golden transcription from several manual transcriptions that were obtained for the recording. Additionally it is also possible to add manual transcriptions via external job at CrowdFlower.

The third responsibility is to provide export of transcribed recordings. This can be done either by downloading archive from the web or by using Torrent.

# Future work

- Since manual transcription of recordings is expensive it would be good to make users transcribe only parts of the recordings in which ASR system wasn't confident enough

  **TODO:** cite (http://www.phontron.com/paper/sperber14slt.pdf)

  . This idea could be used for both user transcription and CrowdFlower transcription.

- With manually transcribed recordings from CloudASR platform it is possible to continuously improve accuracy of the underlying ASR system by adapting the language model to the type of language that the users of the CloudASR really use. Thus CloudASR could provide an option to automatically update language model when a certain amount of new transcribed recordings was collected.

- Because running CloudASR platform is expensive in terms of costs for a server hosting, it would be good to optimize usage of individual workers so that spare workers are shut down when there is no need for them and new workers are started when the traffic arise. This can be achieved either by providing feedback control based systems

  **TODO:** cite (http://shop.oreilly.com/product/0636920028970.do)

  or by using machine learning techniques.

  **TODO:** cite

- As CloudASR platform provides API for speech recognition, it could also be used for another speech related tasks like Language Identification, Speaker Identification, Voice Activity Detection, etc.

# Bibliography


[1] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio, *Theano: A cpu and gpu math compiler in python*, Proc. 9th Python in Science Conf, 2010, pp. 1–7.

[2] Martin Fowler, *Inversion of control containers and the dependency injection pattern*, 2004.

[3] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design patterns: Abstraction and reuse of object-oriented design*, Springer, 1993.

[4] Carl Hewitt, *Viewing control structures as patterns of passing messages*, Artificial intelligence **8** (1977), no. 3, 323–364.

[5] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy H Katz, Scott Shenker, and Ion Stoica, *Mesos: A platform for fine-grained resource sharing in the data center.*, NSDI, vol. 11, 2011, pp. 22–22.

[6] Dirk Merkel, *Docker: lightweight linux containers for consistent development and deployment*, Linux Journal **2014** (2014), no. 239, 2.

[7] Ondrej Plátek and Filip Jurcıcek, *Free on-line speech recogniser based on kaldi asr toolkit producing word posterior lattices*, 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2014, p. 108.

[8] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondřej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, et al., *The kaldi speech recognition toolkit*, (2011).

# List of Tables

# List of Abbreviations

# Attachments