CloudASR: A Web Platform for ASR

Ondrej Klejch

16.3.2015

Motivation

- Easier onboarding of new students
- Sharing of our knowledge
- Data acquisition

Motivation

- Many languages uncovered by Google ASR
  - Google supports 39 languages
- We don't want to compete with Google (yet)
- We can compete on specific domains

Features

- Online and Batch ASR mode
- Multiple languages
- Annotation Interface
- Crowdsourcing
- Easy deployment
- Easy scalability

Batch vs. Online Mode

- Batch Mode:
  - Record recording and send it to the server
  - Receive n-best list
- Online Mode:
  - Send chunks of the recording to the server
  - Receive best path

Batch vs. Online Mode

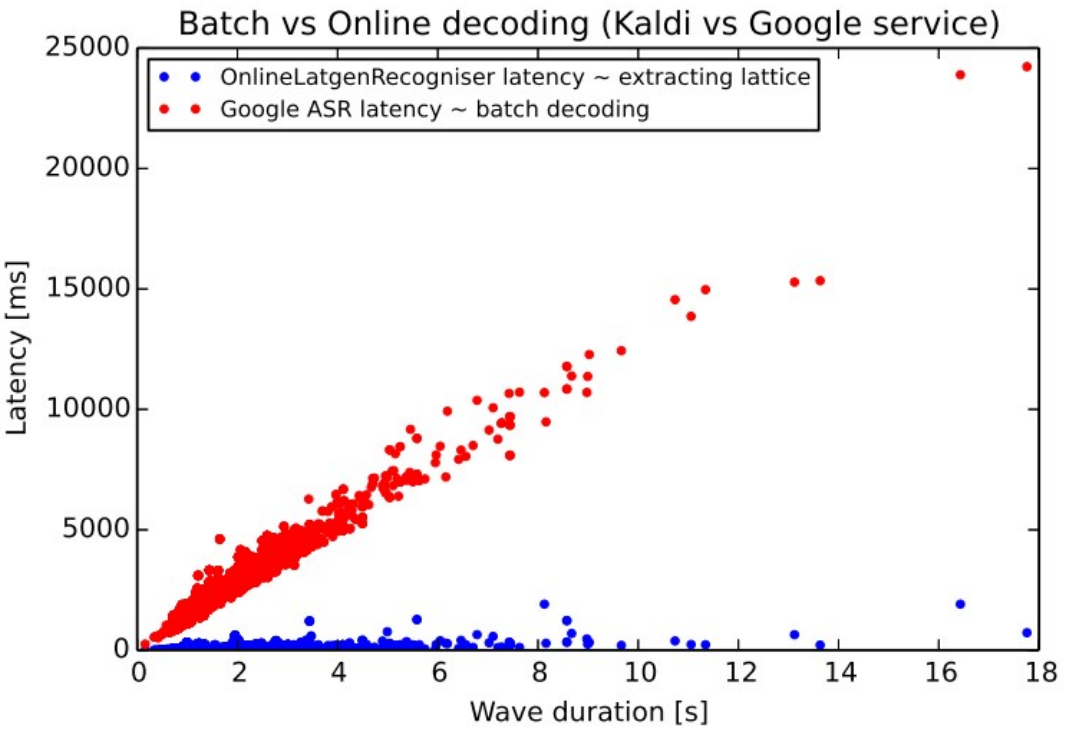
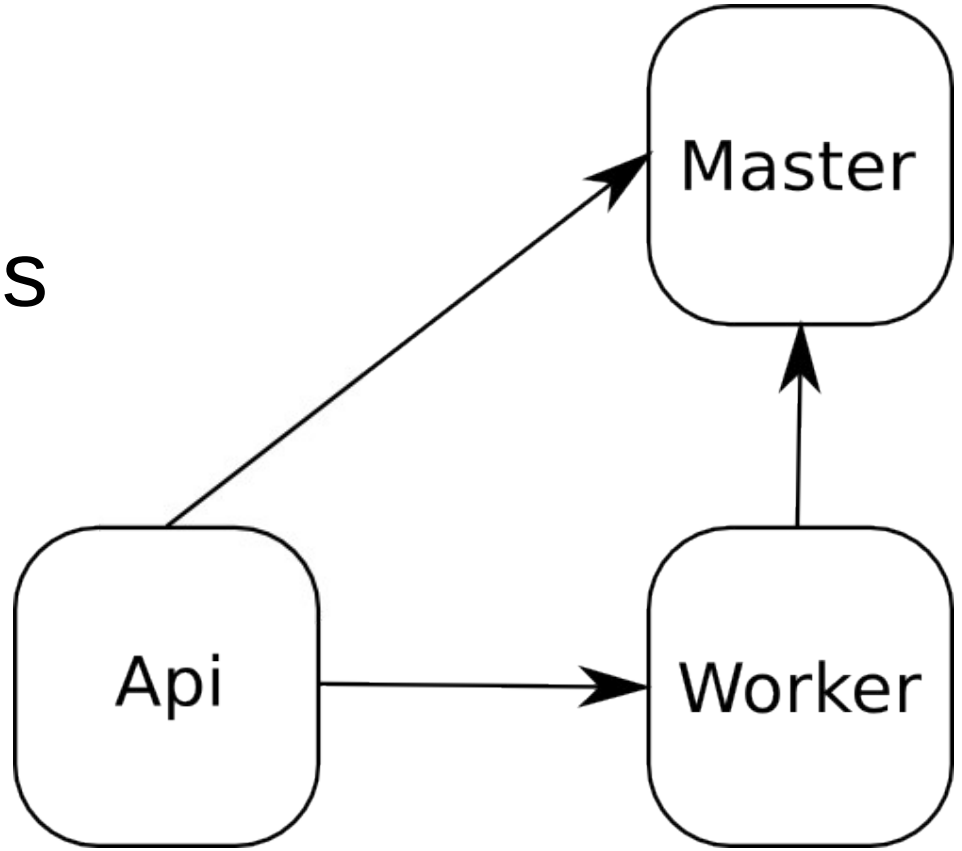


Fig. 3. Relation between latency and utterance length. Comparing on-line decoder (OnlineLatgenRecogniser) and batch decoding (Google cloud ASR service).

Plátek, Ondřej, and Filip Jurčíček. "*Integration of an On-line Kaldi Speech Recogniser to the Alex Dialogue Systems Framework.*" In Text, Speech and Dialogue, pp. 603-610. Springer International Publishing, 2014.

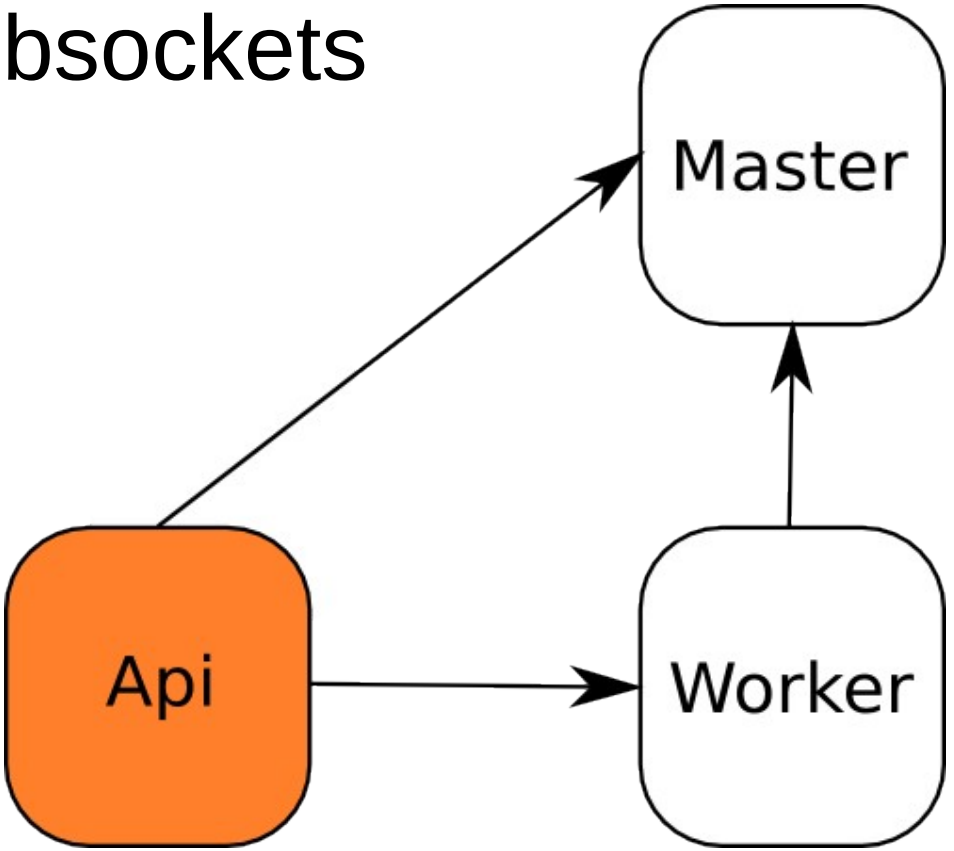
Architecture

- Built from smaller blocks:
  - Master
  - Worker
  - API Frontend
- Blocks send messages



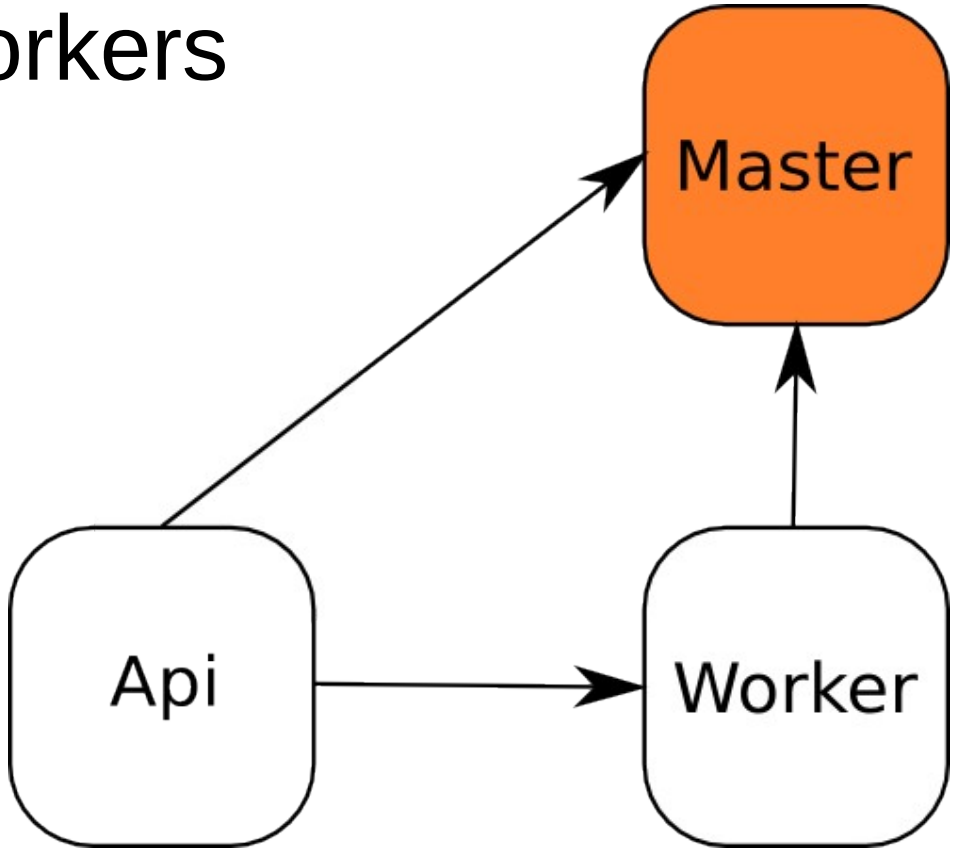
API Frontend

- Receives requests from clients
- Batch mode uses HTTP POST method
- Online mode uses Websockets



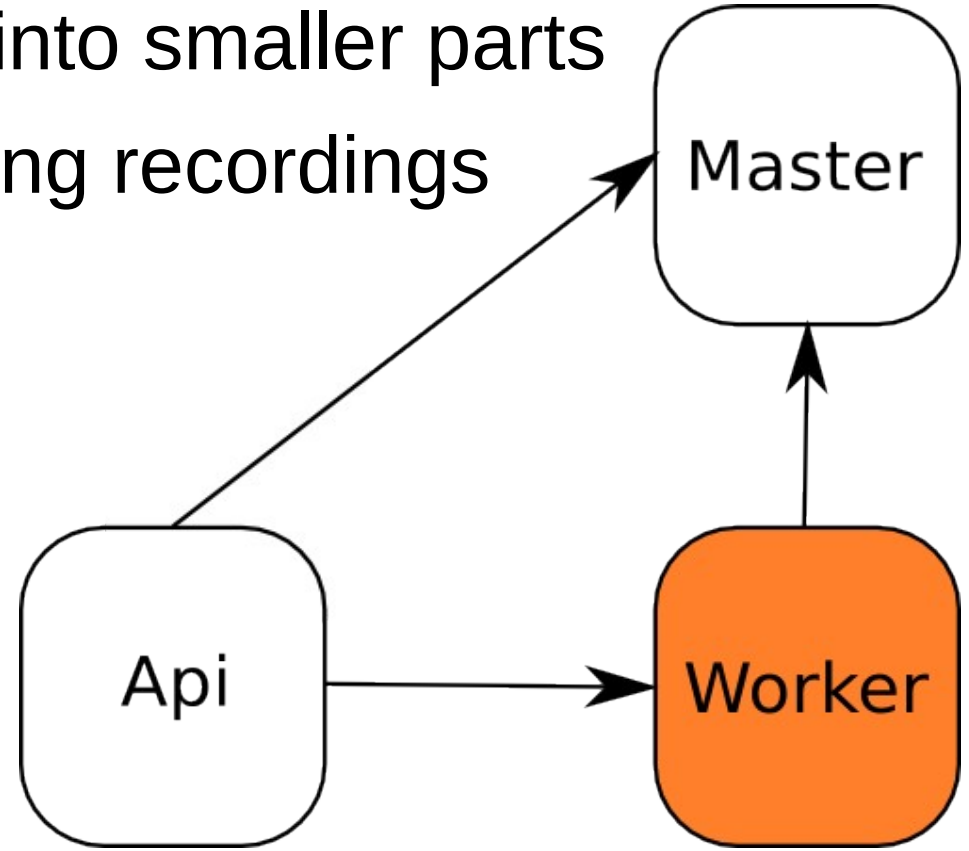
Master

- Receives heartbeats from workers
- Keeps track about running workers
- Distributes tasks to workers



Worker

- Built on top of Pykaldi
- Uses theano neural network for VAD
  - VAD splits recordings into smaller parts
  - We can handle very long recordings



Batch Mode Workflow

- Client sends wav to API Frontend
- API Frontend asks Master for Worker address
- API Frontend sends wav to Worker
- Worker processes the wav
- Worker returns nbest list hypotheses to API Frontend
- API Frontend sends response to Client

Online Mode Workflow

- Client sends chunk to API Frontend
- API Frontend asks Master for Worker address
- Repeat:
  - API Frontend sends chunk to Worker
  - Worker processes the chunk
  - Worker returns best path hypothesis to API Frontend
  - API Frontend sends response to Client
  - API Frontend waits for next chunk

Deployment

Traditional approach:

- Install dependencies on each machine
- Start/stop applications manually
- Make sure that you use same libraries in DEV and PROD

Docker approach:

- A portable, lightweight application runtime and packaging tool.
- Create image with installed dependencies
- Use this image on each machine
- Guaranteed same environment in DEV and PROD

Traditional example

```
apt-get install -y python python-pip

pip install flask flask-socketio

cp -R . /opt/app

cd /opt/app

python run.py
```

Docker usage example

cloudasr/api/Dockerfile

```
FROM ubuntu:14.04

RUN apt-get install -y python python-pip

RUN pip install flask flask-socketio

ADD . /opt/app

WORKDIR /opt/app

CMD python run.py
```

Docker usage example

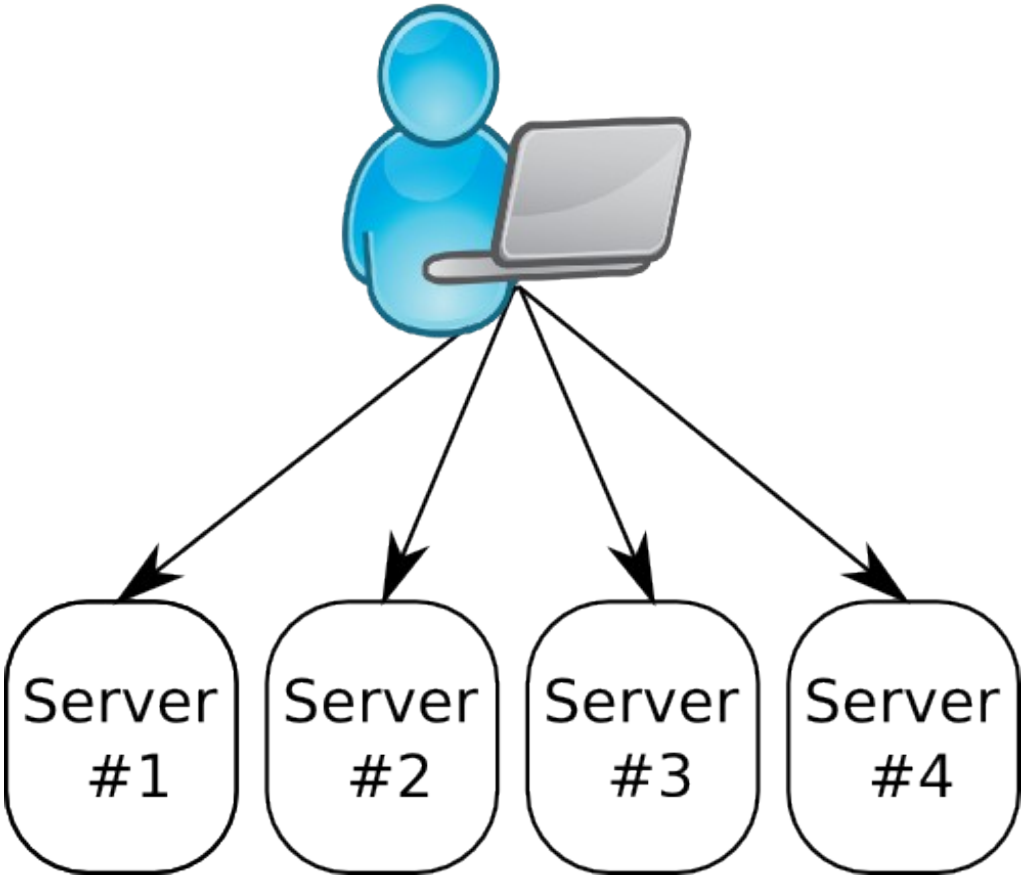
```
OPTS=--name api \
        -p 8080:80 \
        -e MASTER_ADDR=172.17.42.1:8001 \
        -v cloudasr/api:/opt/app

docker build -t ufaldsg/cloud-asr-api cloudasr/api
docker run $OPTS -d ufaldsg/cloud-asr-api
docker stop
```

Cluster deployment

Traditional approach

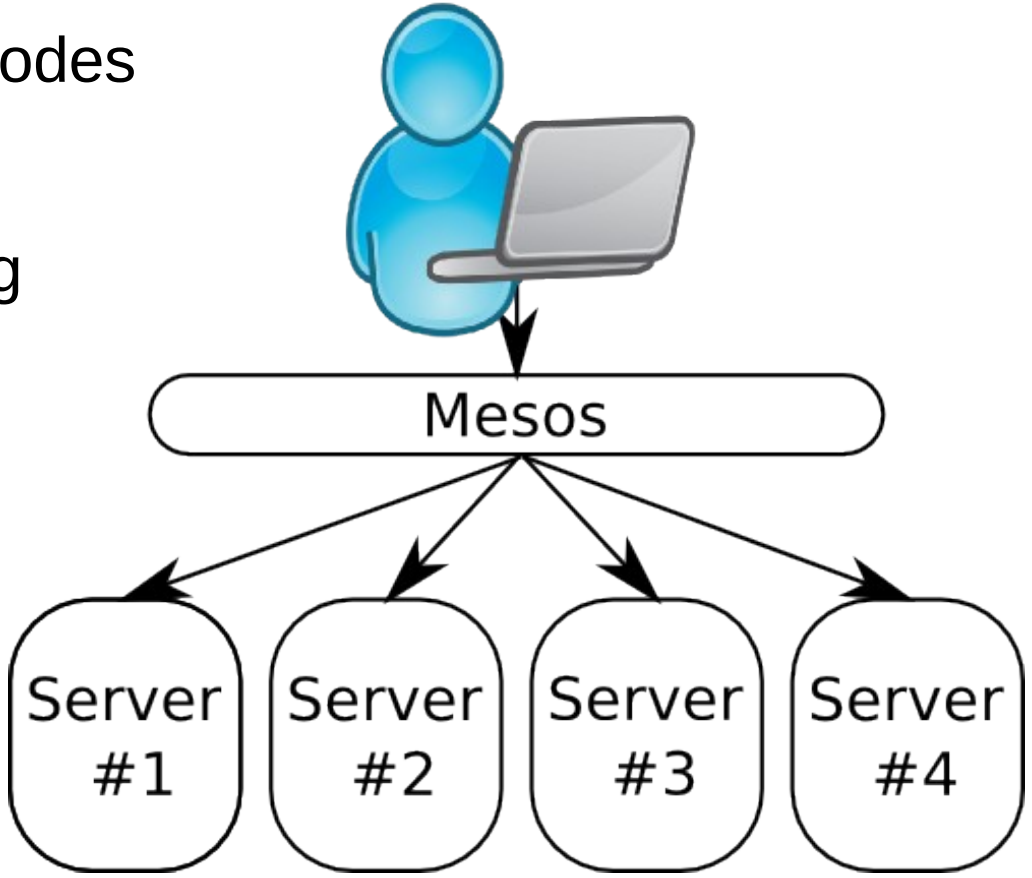
- Setup every machine via ssh
- Run processes manually
- Handle failures manually



Cluster deployment

Mesos approach

- Mesos lets you program against your datacenter like it's a single pool of resources
- Scalability to 10,000s of nodes
- Mesos uses Docker
- Mesos handles scheduling and failure recovery



Marathon API example

api.json

```
{
        "id": "api",
        "container": {
            "type": "DOCKER",
            "docker": { "image": "ufaldsg/cloud-asr-api", }
        },
        "instances": 1, "cpus": 0.25, "mem": 256,
        "env": { "MASTER_ADDR": "tcp://cloudasr_master:31100" },
  }
```

curl -X POST -d @api.json marathon:8080/v2/apps

Marathon API example

api.json

```
{
        "id": "api",
        "container": {
            "type": "DOCKER",
            "docker": { "image": "ufaldsg/cloud-asr-api", }
        },
        "instances": 5, "cpus": 0.25, "mem": 256,
        "env": { "MASTER_ADDR": "tcp://cloudasr_master:31100" },
  }
```

```
curl -X PUT -d @api.json marathon:8080/v2/apps/api
```

Future Work

- Own language model upload
- Language model adaptation
- Acoustic model adaptation

Thank you for your attention!

http://demo.cloudasr.com

https://github.com/UFAL-DSG/cloud-asr