SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Diplomski studij

SOCKET KOMUNIKACIJA POMOĆU ZEROMQ BIBLIOTEKE

Marko Grubeša

Osijek, 2020.

Sadržaj

1.Uvod

Izrada programskih rješenja danas često zahtijeva korištenje različitih tehnologija. Mnogi programski jezici koriste se za izradu jednog projekta te oni moraju znati međusobno komunicirati. Te skripte mogu komunicirati na puno načina, a jedan od njih je korištenje socketa.

ZeroMQ biblioteka olakšava socket komunikaciju te će se ovaj seminar temeljiti na korištenju te biblioteke. Ostvarit će se komunikacija između node.js i python skripte, tako što će node.js skripta slati sliku, python će ju obraditi te poslati nazad poruku.

2.ZeroMQ

ZeroMQ je asinkrona biblioteka visokih performansi, namijenjena upotrebi u distribuiranim ili istodobnim aplikacijama. Omogućuje red poruka, ali za razliku od posredničkog softvera orijentiranog na poruke, sustav ZeroMQ može se izvoditi bez namjenskog posrednika za poruke. ZeroMQ podržava uobičajene obrasce slanja poruka (pub / sub, zahtjev / odgovor, klijent / poslužitelj i drugi) tijekom različitih prijenosa (TCP, proces, inter-proces, multicast, WebSocket i više), čineći međuprocesne poruke jednostavnim. Ovo čini kod jasnim, modularnim i izuzetno lakim za skaliranje. Mnogo tehnologija koristi ZeroMQ biblioteku te je popularan u mnogim programskim jezicima kao što su C#, Java i mnogi drugi. ZeroMQ API pruža sockete, od kojih svaka predstavlja "many-to- many" vezu između krajnjih točaka te one zahtijevaju korištenje određenih obrazaca pri slanju poruke. Osnovni ZeroMQ obrasci su:

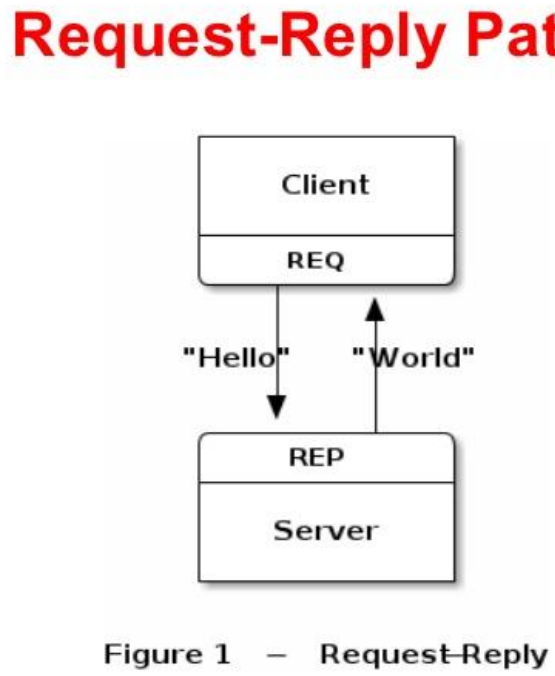
Zatjev-odgovor, on povezuje skup klijenata s nizom usluga.

Objava-pretplata, povezuje izdavače s pretplatnicima. Ovo je obrazac distribucije podataka.

Push-pull(cjevovod), povezuje čvorove u uzorku " fan-out / fan-in" koji mogu imati više koraka i petlje. Ovo je paralelni obrazac raspodjele i skupljanja zadataka.

Ekskluzivni par, povezuje dva socketa u ekskluzivnom paru. (Ovo je napredni model niske razine za određene slučajeve upotrebe.).

Programsko rješenje koristit će Zahtjev-odgovor (Request-Reply) obrazac.



Slika 2.1 Prikaz Request-reply obrasca

3.Ostale Tehnologije

3.1 Node.js

Platforma koja služi za izradu brzih real-time aplikacija i koja je napravljena pomoću JavaScript programskog jezika naziva se Node.js. Node.js implementiran je 2009. godine s ciljem da omogući korisniku skriptiranje web stranice. Daljnjim razvojem platforme, glavna funkcija s korisničkog skriptiranja prešla je na skriptiranje od strane poslužitelja. S time se postiglo brže učitavanje stranica jer se sadržaj učitava prvo na poslužitelju. Poslužitelj preuzima informacijsko opterećenje te zbog toga korisničko računalo brže učitava web sadržaj. Node.js predstavlja paradigmu "JavaScript posvuda", objedinjujući razvoj web aplikacija oko jednog programskog jezika, umjesto različitih jezika za skripte na strani poslužitelja i klijenta.

```
var http = require('http');

http.createServer(function (req, res) {
   res.writeHead(200, {'Content-Type': 'text/plain'});
   res.end('Hello World!');
}).listen(8080);
```

Slika 3.1 Primjer Node.js skripte

3.2 Python

Python je interpretirani programski jezik visokog nivoa opće namjene. Napravio ga je Guido van Rossum i prvi put je objavljen 1991. godine. Python-ova filozofija dizajna naglašava čitljivost koda uz značajno korištenje prostora. Njegovi jezični konstrukti i objektno orijentirani pristup nastoje pomoći programerima u pisanju jasnog i logičnog koda za male i velike projekte.

Python se dinamički tipka i ima „garbage collector“ te se programer ne mora brinuti o oslobađanju memorije. Podržava više programskih paradigmi, uključujući strukturirano (posebno proceduralno), objektno orijentirano i funkcionalno programiranje. Python se često opisuje kao jezik s "uključenim baterijama" zbog njegove sveobuhvatne standardne biblioteke. Koristi se pretežno u strojnom učenju, obradi slike i u mnogim drugim područjima.

```
      # Python3 program to add two numbers
```

```
num1 = 15
num2 = 12

# Adding two nos
sum = num1 + num2

# printing values
print("Sum of {0} and {1} is {2}" .format(num1, num2, sum))
```

Slika 1.2 Primjer Python skripte

4. Programsko rješenje

Programsko rješenje prikazuje komunikaciju python i node.js skripte.

```
const zmq = require("zeromq");
const { promisify } = require("util");
const fs = require("fs");

readFileAsync = promisify(fs.readFile);
readDirAsync = promisify(fs.readdir);

async function run() {
  const sock = new zmq.Request();

  sock.connect("tcp://127.0.0.1:5555");
  console.log("Producer bound to port 5555");

  files = await readDirAsync("images/");
  for (var i = 0; i < files.length; i++) {
    img = await readFileAsync("images/" + files[i]);
    await sock.send(img);
    console.log("send request");
    const [result] = await sock.receive();
    console.log(JSON.parse(result));
  }
}

run();
```

Slika 4.1 Node.js skripta rješenja

Iz slike 4.1 vidi se da se pomoću require uključuju ZeroMQ bilioteka te se još uključuju util i fs moduli, koji će nam služiti za čitanje direktorija i datoteka. Unutar asinkrone funkcije run, kreira se novi Request, koji se spaja na port 5555.

Pomoću readdirAsync funkcije čitamo direktorij gdje se nalaze slike te prolaskom kroz petlju i pomoću funkcije readFileAsync slike se šalju python skripti gdje će se one dalje obrađivati.

```
import numpy as np
import zmq
import base64
import json
from json import JSONEncoder
import io
from skimage.feature import blob_dog
```

```
from PIL import Image

def set_up_host():
    context = zmq.Context()
    socket = context.socket(zmq.REP)
    socket.bind("tcp://*:5555")
    return socket



def send_to_client(socket):
    request = socket.recv()
    if request:
        print("Received request")
        img = None
        err = None
        out_str="Image error,not a valid image"
        try:
            buf = Image.open(io.BytesIO(request)).convert('LA')##pretvara
u gray scale

            img = np.array(buf)

        except Exception as e:
            print("Image file not understood!")
            err = e
        if img is not None:
            err="All is okay"
            pass
            out_dict = {"error" : err, "output" : img.shape}


            out_str = json.dumps(out_dict,cls=NumpyArrayEncoder)

        socket.send_json(out_str)
        print("send respond")

class NumpyArrayEncoder(JSONEncoder):
    def default(self, obj):
        if isinstance(obj, np.ndarray):
            return obj.tolist()
        return JSONEncoder.default(self, obj)

if __name__ == "__main__":
    socket = set_up_host()
    while True:
        try:
            send_to_client(socket)
        except KeyboardInterrupt:
            break
```

Slika 4.2 Python skripta rješenja

Iz slike 4.2 prvotno se uključuju sve potrebne biblioteke koje će se koristiti. U funkciji set_up_host() socket se veže na port 5555. Unutar funkcije send_to_client() request se prima te dobivenu slike pretvara u gray-scale sliku.

Nakon toga se provjerava je li slika ispravna, ako je neispravna šalje se određena poruka te se podiže iznimka. Ako je sve u redu, šalje se json poruka zajedno sa dimenzijom obrađene slike. Klasa NumpyArrayEncode omogućuje slanje polja kao JSON poruku.

```
C:\Program Files\nodejs\node.exe zmq-client.js
Producer bound to port 5555
send request
{"error": "All is okay", "output": [1024, 1224, 2]}
send request
{"error": "All is okay", "output": [225, 225, 2]}
send request
Image error,not a valid image
```

Slika 4.3 Primljene poruke od python skripte

```
========== RESTART: C:\Users\Marko\Desktop\sp-seminar\zmq_image.py ==========
Received request
send respond
Received request
send respond
Received request
Image file not understood!
send respond
```

Slika 4.4 Python skripta potvrđuje primitak slike i šalje nazad poruku

5.Zaključak

Cilj ovog rada bio je pokazati komunikaciju između python i node.js skripte. Za to je korištena socket komunikacija pomoću ZeroMQ biblioteke. Programsko rješenje prikazuje kako node.js skripta čita direktorij i šalje slike python skripti. Python skripta obrađuje sliku te šalje nazad poruku u obliku JSON stringa. Za sve to je korišten Request reply obrazac iz ZeroMQ biblioteke, gdje node.js predstavlja client stranu, a python server stranu.

6.Literatura

1. https://zeromq.org/get-started -Datum pristupa linku: 8.6.2020
2. https://zeromq.org/socket-api/ -Datum pristupa linku: 8.6.2020
3. https://nodejs.org/en/ -Datum pristupa linku: 8.6.2020
4. https://en.wikipedia.org/wiki/Python_(programming_language) -Datum pristupa linku: 8.6.2020