

CDN Latency

CDN Latency

Chris, Nilou and Andres

University of California, Riverside

May 27, 2016

Outline

1 Introduction

2 Methodology

- CDNs collection
- Choosing candidates
- Collecting ping data
- Building the graph
- Visualization

3 Results

Outline

1 Introduction

2 Methodology

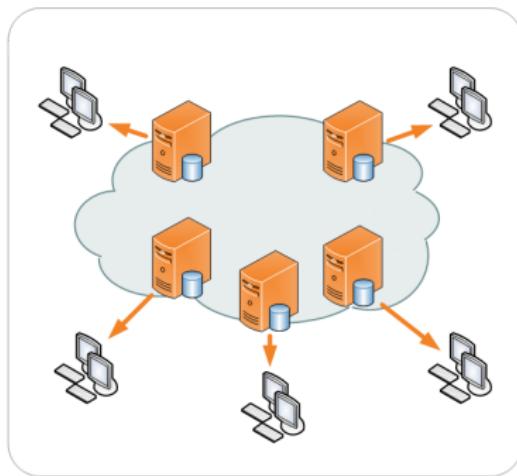
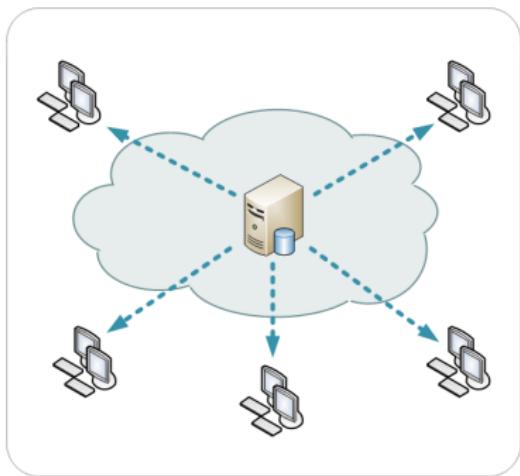
- CDNs collection
- Choosing candidates
- Collecting ping data
- Building the graph
- Visualization

3 Results

Introduction

- Measuring and Evaluating Large-Scale CDNs (Huang et al., 2008).
- The goal of a CDN is to serve content to end-users with high availability and high performance.

Introduction



Introduction

- But... Where do we deploy a new CDN server???

Outline

1 Introduction

2 Methodology

- CDNs collection
- Choosing candidates
- Collecting ping data
- Building the graph
- Visualization

3 Results

Overview

- ① Collect information about current CDNs
- ② Select a group of possible candidates
- ③ Measure the latency between points (Ping time)
- ④ Graph analysis
- ⑤ Visualization

Outline

1 Introduction

2 Methodology

- CDNs collection
- Choosing candidates
- Collecting ping data
- Building the graph
- Visualization

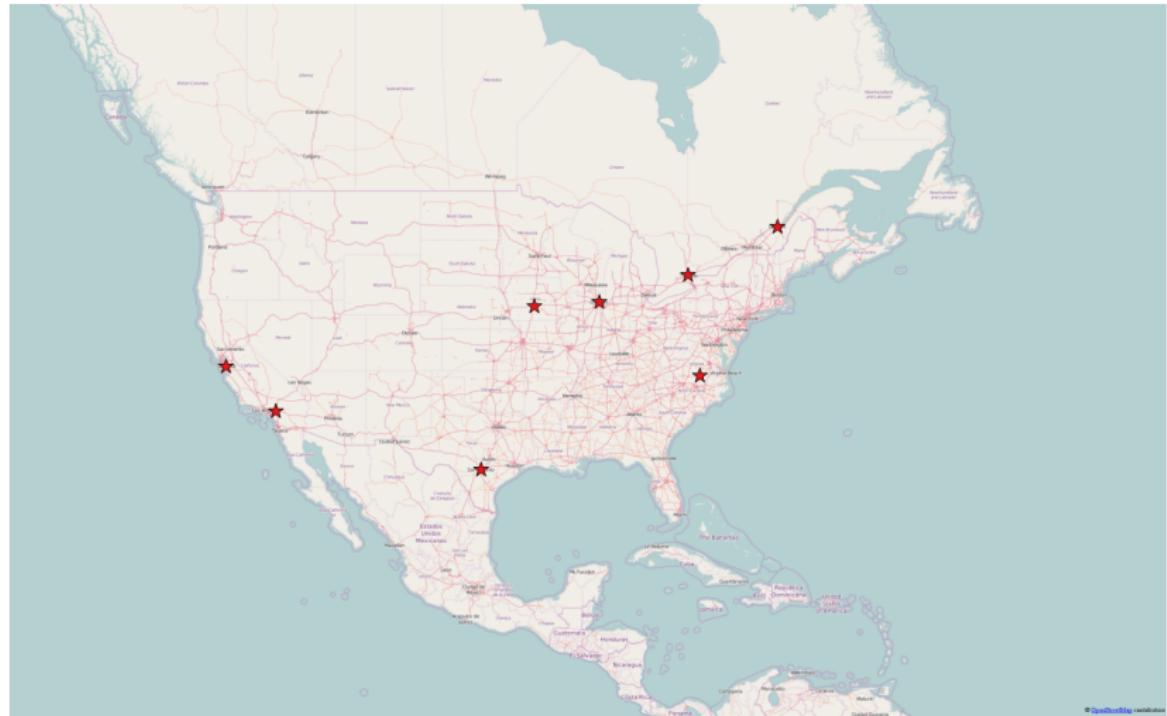
3 Results

Virtual machines

- Microsoft Azure: Cloud Computing Platform & Services ¹
- 8 instances in different geographical regions
 - ① East USA
 - ② Central USA
 - ③ South Central USA
 - ④ West USA
 - ⑤ North Central USA
 - ⑥ East Canada
 - ⑦ Central Canada
 - ⑧ Riverside

¹<https://azure.microsoft.com/>

Virtual machines



Outline

1 Introduction

2 Methodology

- CDNs collection
- Choosing candidates
- Collecting ping data
- Building the graph
- Visualization

3 Results

Choosing candidates

- Universities Worldwide ²
- Universities from USA and Canada

²<http://univ.cc/>

Choosing candidates

- Cleaning and processing
 - Remove urls which did not respond to ping
 - Remove duplicates (in the same city)
 - Remove incorrect format
 - Extract latitude and longitude (reverse geocoding)
- End up with 97 candidates

Python scripts

```
import xml.dom.minidom
import socket
from geoip import geolite2
import subprocess

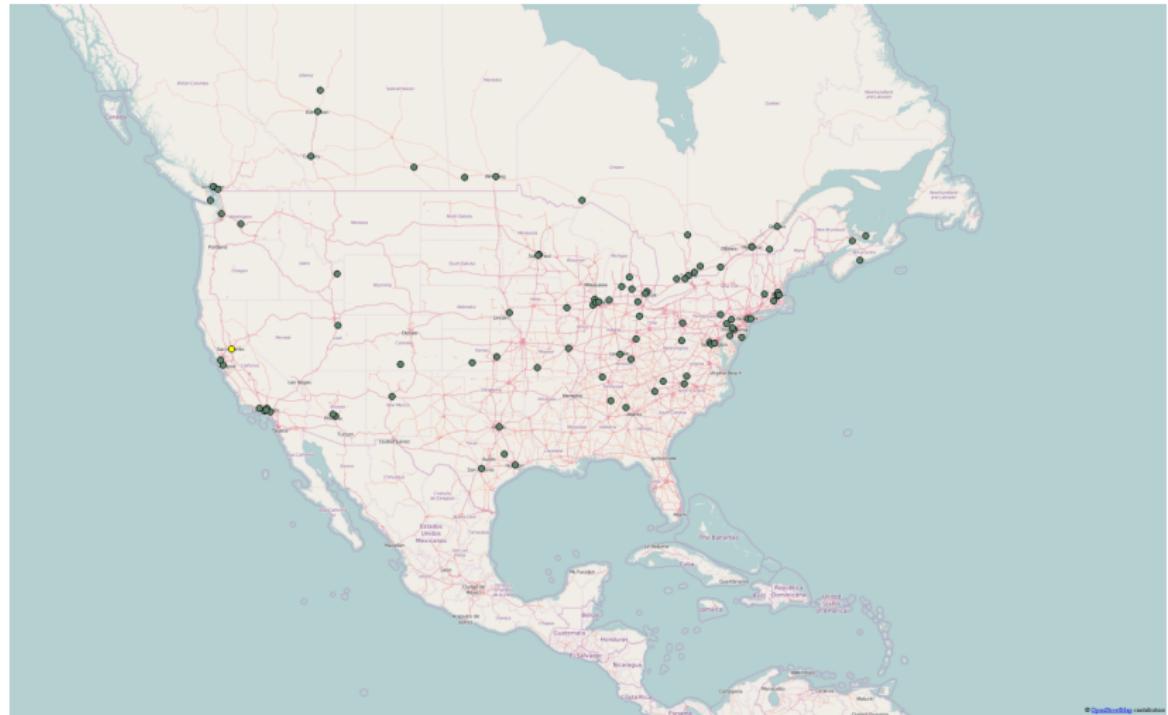
doc = xml.dom.minidom.parse("usa.xml")
list = doc.getElementsByTagName("a")
for item in list:
    url = item.getAttribute("href")[7:-1]
    try:
        addr = socket.gethostname(url)
        match = geolite2.lookup(addr)
    except socket.error, msg:
        addr
    else:
        try:
            output = subprocess.check_output(["ping", "-c1", url])
        except subprocess.CalledProcessError,e:
            url
        else:
            print "{0}".format(url)
```

Python scripts

```
import socket
from geoip import geolite2
import subprocess
import geocoder
import time

f = open('dns3.txt', 'r')
for url in f.readlines():
    try:
        url = url[:-1]
        addr = socket.gethostbyname(url)
        match = geolite2.lookup(addr)
        lat = round(match.location[0],2)
        lon = round(match.location[1],2)
        g = geocoder.google([lat,lon], method='reverse')
        time.sleep(2)
    except socket.error, msg:
        print ""
    else:
        print(url+";"+addr+";"+str(lat)+"."++str(lon)+";"+g.city+";"+g.state)
```

Choosing candidates



Outline

1 Introduction

2 Methodology

- CDNs collection
- Choosing candidates
- **Collecting ping data**
- Building the graph
- Visualization

3 Results

Ping data

- We collect ping data from each current CDN to all the candidates
- Run a script at each virtual machine
- Transmit 5 packages and extract the average ping time

Python scripts

```
import subprocess

f = open('dns3.txt', 'r')
for url in f.readlines():
    try:
        url = url[:-1]
        output = subprocess.check_output(["ping", "-c5", url])
    except subprocess.CalledProcessError,e:
        url
    else:
        ping = float(output.split('\n')[-2:-1][0].split('/')[-2:-1][0])
        print "{0};{1}".format(url, ping)
```

Outline

1 Introduction

2 Methodology

- CDNs collection
- Choosing candidates
- Collecting ping data
- Building the graph**
- Visualization

3 Results

Building the graph

- Merge results for each virtual machine (CDN)
- Use networkX package in python
- Final graph has 105 nodes and 776 edges

Python scripts

```
import networkx as nx
import pandas as pd

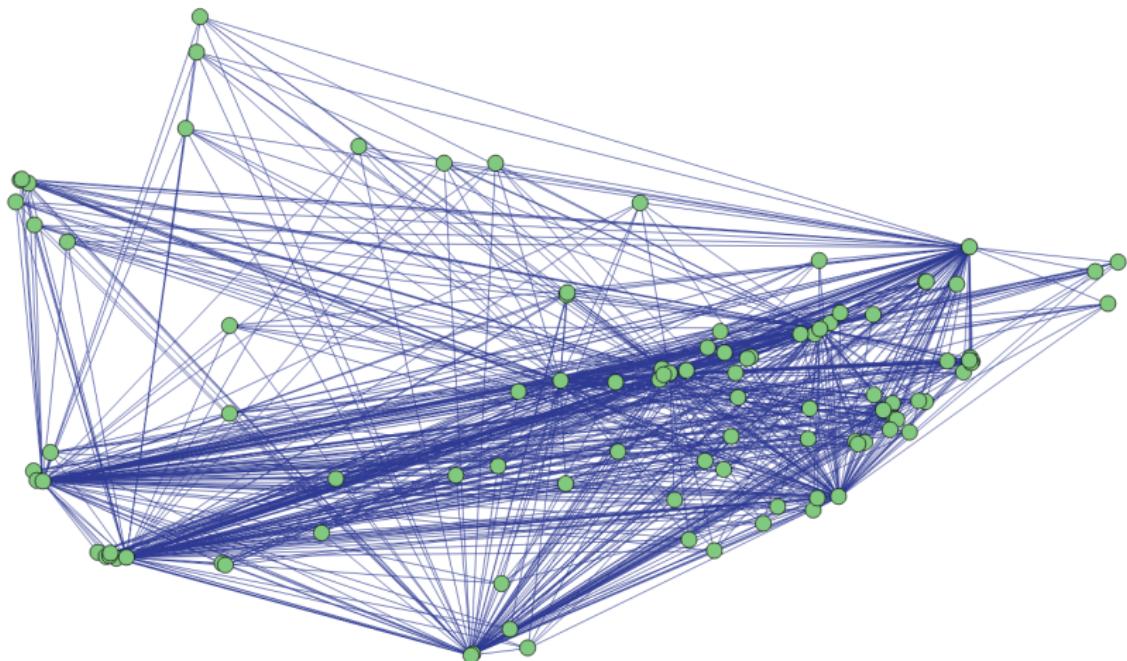
G=nx.Graph()
## Riverside
center_y = 33.9680
center_x = -117.3194
center = (center_x,center_y)
G.add_node(center)

candidates = pd.read_csv('candidates.csv', sep=';', header=None
                         , names=['url','ip','lat','lon','city','state'])
pings = pd.read_csv('pings.csv', sep=';', header=None, names=['url','ping'])

nodes = candidates.loc[:,['url','lat','lon']].merge(pings).loc[:,['lat','lon','ping']]

for index, node in nodes.iterrows():
    point = (node['lon'],node['lat'])
    G.add_node(point)
    G.add_edge(center,point,weight=node['ping'],cdn='Riverside')
```

Building the graph



Outline

1 Introduction

2 Methodology

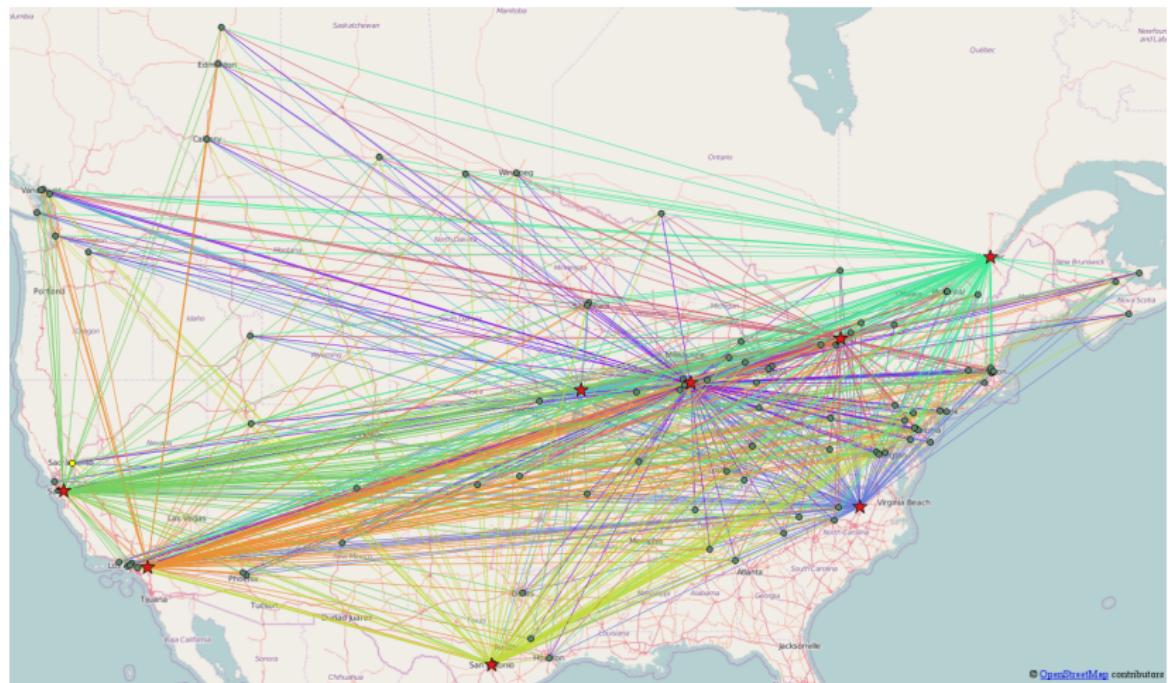
- CDNs collection
- Choosing candidates
- Collecting ping data
- Building the graph
- Visualization

3 Results

Visualization

- Use networkX to save the output to a shapefile format
- Use QGIS to visualize data

Visualization



Outline

1 Introduction

2 Methodology

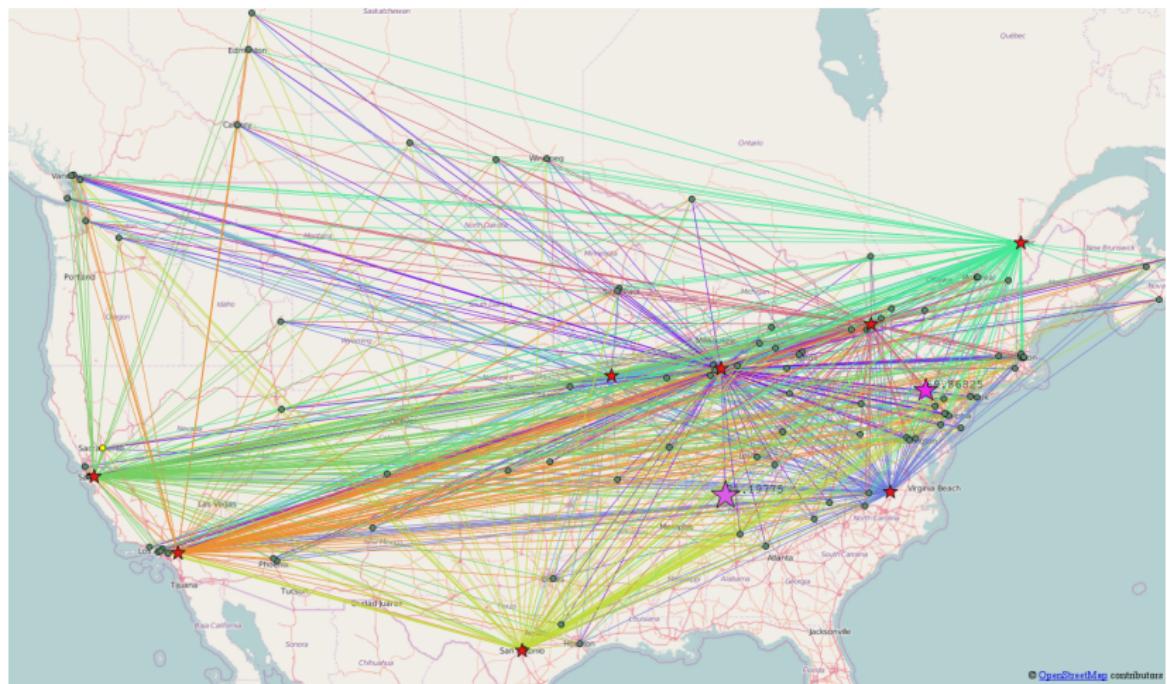
- CDNs collection
- Choosing candidates
- Collecting ping data
- Building the graph
- Visualization

3 Results

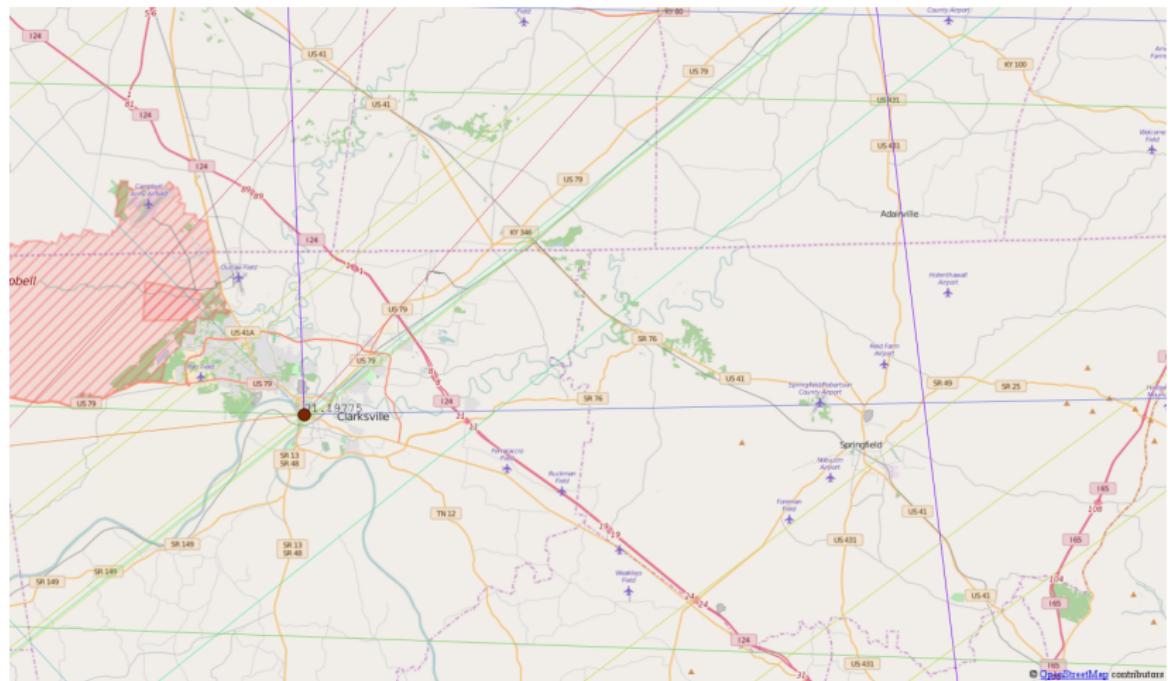
Results

- Iterate at each candidates to all the CDNs
- Compute the average ping time
- Sort by average time ascendantly

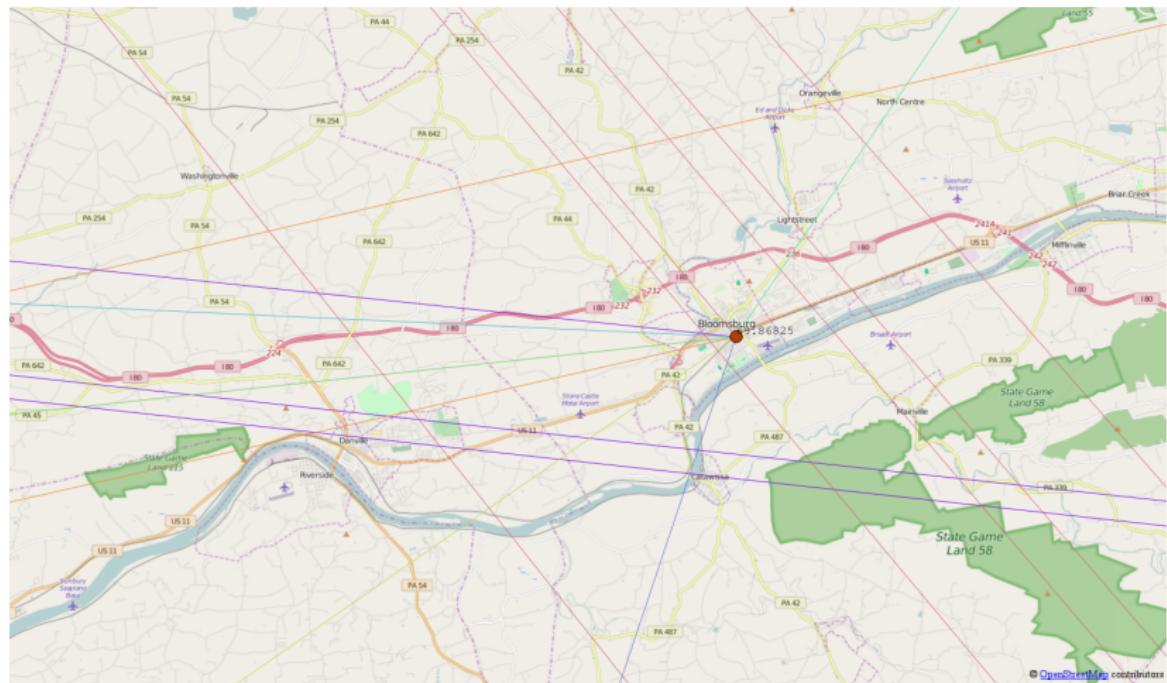
Results



Results



Results



Thank you!!!