In [1]:
```
import numpy as np
import json
import matplotlib.pyplot as plt
from tqdm import tqdm
import random
import subprocess
import time
import pandas as pd
import os
```

Read the data

In [2]:
```
with open("webpacks/webpack_bodies.json") as file:
    data = file.read()

splitted_data = data.split('{"page"')[1:]
splitted_data = [json.loads('{"page"' + line)["body"] for line in s
plitted_data]

for i in range(50):
    name = "0" * (12 - len(str(i))) + str(i)
    with open("webpacks/webpack_bodies_" + name + ".json") as file:
        data = file.read().split('{"page"')[1:]
        splitted_data += [json.loads('{"page"' + line)["body"] for
line in data]
```

Perform compression for bundled approach

In [17]:
```
def log(file, msg):
    f = open(file, 'a+')
    f.write(msg + '\n')
    f.close()

def get_seconds(time):
    min_ind = time.find('m')
    mins = int(time[:min_ind])
    second = float(time[min_ind + 1:-1])
    return mins * 60 + second
```

In [219]:
```
rates_gzip_bundled = []
rates_brotli_bundled = []
times_gzip_bundled = []
times_brotli_bundled = []
speed_gzip_bundled = []
speed_brotli_bundled = []
init_sizes_bundled = []
for i in range(4000):
    rates_gzip_compressed = []
    rates_brotli_compressed = []
    times_gzip_compressed = []
    times_brotli_compressed = []
    speed_gzip_compressed = []
    speed_brotli_compressed = []

    # write the text of a bundle to file to use it for compression
later
    with open("example.txt", "w") as file:
        file.write(splitted_data[i])
    size_non_compressed = os.stat("example.txt").st_size
    init_sizes_bundled.append(size_non_compressed)

    # do the gzip compression with different levels
    for level in range(4, 10):
        result = subprocess.run(["bash", "gzip_compress.sh", str(le
vel), "time.txt",
                                 "example_gzip.txt.gz", "example.tx
t"])
        #previous script saves the time into the file
        with open("time.txt") as file:
            user_sys = file.read().strip().split('\n')[1:]
        time = get_seconds(user_sys[0].split('\t')[1]) + get_second
s(user_sys[1].split('\t')[1])
        size_gzip_compressed = os.stat("example_gzip.txt.gz").st_si
ze
        rates_gzip_compressed.append(size_non_compressed / size_gzi
p_compressed)
        times_gzip_compressed.append(time)
        speed_gzip_compressed.append(size_non_compressed / time)

    # do the brotli compression with different levels
```

```
    for level in range(4, 12):
        result = subprocess.run(["bash", "brotli_compress.sh", str(
level), "time.txt",
                                 "example_brotli.txt.br", "example.
txt"])
        with open("time.txt") as file:
            user_sys = file.read().strip().split('\n')[1:]
        time = get_seconds(user_sys[0].split('\t')[1]) + get_second
s(user_sys[1].split('\t')[1])
        size_br_compressed = os.stat("example_brotli.txt.br").st_si
ze
        rates_brotli_compressed.append(size_non_compressed / size_b
r_compressed)
        times_brotli_compressed.append(time)
        speed_brotli_compressed.append(size_non_compressed / time)

    rates_gzip_bundled.append(rates_gzip_compressed)
    rates_brotli_bundled.append(rates_brotli_compressed)
    times_gzip_bundled.append(times_gzip_compressed)
    times_brotli_bundled.append(times_brotli_compressed)
    speed_gzip_bundled.append(speed_gzip_compressed)
    speed_brotli_bundled.append(speed_brotli_compressed)

    if i != 0 and i % 50 == 0:
        log("logs.txt", "rates_gzip: " + str(np.mean(rates_gzip_bun
dled, axis=0)))
        log("logs.txt", "rates_brotli: " + str(np.mean(rates_brotli
_bundled, axis=0)))
        log("logs.txt", "times_gzip: " + str(np.mean(times_gzip_bun
dled, axis=0)))
        log("logs.txt", "times_brotli: " + str(np.mean(times_brotli
_bundled, axis=0)))
        log("logs.txt", "speed_gzip: " + str(np.mean(speed_gzip_bun
dled, axis=0)))
        log("logs.txt", "speed_brotli: " + str(np.mean(speed_brotli
_bundled, axis=0)))
```

In [408]:
```
frame = pd.DataFrame()
frame["name"] = ["gzip 4", "gzip 5", "gzip 6", "gzip 7", "gzip 8",
"gzip 9",
                 "brotli 4", "brotli 5", "brotli 6", "brotli 7", "b
rotli 8", "brotli 9", "brotli 10", "brotli 11"]

frame["rates"] = np.hstack((np.mean(rates_gzip_bundled, axis=0), np
.mean(rates_brotli_bundled, axis=0)))
frame["savings"] = 1 - 1 / np.hstack((np.mean(rates_gzip, axis=0),
np.mean(rates_brotli_bundled, axis=0)))
frame["speed(MB/s)"] = np.hstack((np.mean(speed_gzip_bundled, axis=
0), np.mean(speed_brotli_bundled, axis=0))) / 1000000

frame
```

Out[408]:

	name	rates	savings	speed(MB/s)
0	gzip 4	3.260182	0.726951	20.650285
1	gzip 5	3.357855	0.736465	17.320770
2	gzip 6	3.392349	0.739789	14.828584
3	gzip 7	3.402581	0.740678	13.650937
4	gzip 8	3.409667	0.741257	11.943546
5	gzip 9	3.410150	0.741290	11.749338
6	brotli 4	3.460357	0.711012	16.654836
7	brotli 5	3.714279	0.730769	11.635901
8	brotli 6	3.745654	0.733024	10.270090
9	brotli 7	3.771763	0.734872	7.601850
10	brotli 8	3.782508	0.735625	5.944935
11	brotli 9	3.793822	0.736414	4.489888
12	brotli 10	4.044363	0.752742	1.181333
13	brotli 11	4.118859	0.757214	0.519743

Unbundled approach

In [214]:
```
rates_gzip_unbundled = []
rates_brotli_unbundled = []
times_gzip_unbundled = []
times_brotli_unbundled = []
speed_gzip_unbundled = []
speed_brotli_unbundled = []
init_sizes_unbundled = []
```

```
for i in range(600):
    # write the text of a bundle to file to use it for getting chun
ks from bundle later
    with open("third_party/bundle_analyzer/text_bundle.txt", "w") a
s file:
        file.write(splitted_data[i])
    try:
        # save chunks from bundle to parsed_bundle.json file
        result = subprocess.run(["node", "--experimental-modules",
"third_party/bundle_analyzer/get_chunks.js"])
    except:
        continue
    # get chunks
    with open("parsed_bundle.json") as file:
        codes = [line['code'] for line in json.loads(file.read())]

    sizes_gzip_compressed = np.zeros(6)
    sizes_brotli_compressed = np.zeros(8)
    times_gzip_compressed = np.zeros(6)
    times_brotli_compressed = np.zeros(8)
    overall_init_size = 0

    for code in codes:
        if not code:
            continue
        # write the text of a bundle to file to use it for compress
ion later
        with open("example.txt", "w") as file:
            file.write(code)
        overall_init_size += os.stat("example.txt").st_size

        # do the gzip compression with different levels
        for level in range(4, 10):
            result = subprocess.run(["bash", "gzip_compress.sh", st
r(level), "time.txt",
                                     "example_gzip.txt.gz", "exampl
e.txt"])
            with open("time.txt") as file:
                user_sys = file.read().strip().split('\n')[1:]
            time = get_seconds(user_sys[0].split('\t')[1]) + get_se
conds(user_sys[1].split('\t')[1])
            sizes_gzip_compressed[level - 4] += os.stat("example_gz
ip.txt.gz").st_size
            times_gzip_compressed[level - 4] += time

        # do the brotli compression with different levels
        for level in range(4, 12):
            result = subprocess.run(["bash", "brotli_compress.sh",
str(level), "time.txt",
                                     "example_brotli.txt.br", "exam
ple.txt"])
            with open("time.txt") as file:
                user_sys = file.read().strip().split('\n')[1:]
            time = get_seconds(user_sys[0].split('\t')[1]) + get_se
```

```
conds(user_sys[1].split('\t')[1])
            sizes_brotli_compressed[level - 4] += os.stat("example_
brotli.txt.br").st_size
            times_brotli_compressed[level - 4] += time

    rates_gzip_unbundled.append(overall_init_size / sizes_gzip_comp
ressed)
    rates_brotli_unbundled.append(overall_init_size / sizes_brotli_
compressed)
    times_gzip_unbundled.append(times_gzip_compressed)
    times_brotli_unbundled.append(times_brotli_compressed)
    speed_gzip_unbundled.append(overall_init_size / times_gzip_comp
ressed)
    speed_brotli_unbundled.append(overall_init_size / times_brotli_
compressed)
    init_sizes_unbundled.append(overall_init_size)

    if i != 0 and i % 100 == 0:
        log("logs2.txt", "rates_gzip: " + str(np.mean(rates_gzip_un
bundled, axis=0)))
        log("logs2.txt", "rates_brotli: " + str(np.mean(rates_brotl
i_unbundled, axis=0)))
        log("logs2.txt", "times_gzip: " + str(np.mean(times_gzip_un
bundled, axis=0)))
        log("logs2.txt", "times_brotli: " + str(np.mean(times_brotl
i_unbundled, axis=0)))
        log("logs2.txt", "speed_gzip: " + str(np.mean(speed_gzip_un
bundled, axis=0)))
        log("logs2.txt", "speed_brotli: " + str(np.mean(speed_brotl
i_unbundled, axis=0)))
```

In [419]:
```
frame = pd.DataFrame()
frame["name"] = ["gzip 4", "gzip 5", "gzip 6", "gzip 7", "gzip 8",
"gzip 9",
                 "brotli 4", "brotli 5", "brotli 6", "brotli 7", "b
rotli 8", "brotli 9", "brotli 10", "brotli 11"]

frame["rates_bundled"] = np.hstack((np.mean(rates_gzip_bundled[:600
], axis=0),
                                    np.mean(rates_brotli_bundled[:6
00], axis=0)))
frame["savings_bundled"] = 1 - 1 / np.hstack((np.mean(rates_gzip_bu
ndled[:600], axis=0),
                                              np.mean(rates_brotli_
bundled[:600], axis=0)))
frame["speed_bundled(MB/s)"] = np.hstack((np.mean(speed_gzip_bundle
d[:600], axis=0),
                                          np.mean(speed_brotli_bund
led[:600], axis=0))) / 1000000

frame["rates_unbundled"] = np.hstack((np.mean(rates_gzip_unbundled,
axis=0),
                                      np.mean(rates_brotli_unbundle
d, axis=0)))
frame["savings_unbundled"] = 1 - 1 / np.hstack((np.mean(rates_gzip_
unbundled, axis=0),
                                                np.mean(rates_brotl
i_unbundled, axis=0)))
frame["speed_unbundled(MB/s)"] = np.hstack((np.mean(speed_gzip_unbu
ndled, axis=0),
                                            np.mean(speed_brotli_un
bundled, axis=0))) / 1000000
frame
```

Out[419]:

	name	rates_bundled	savings_bundled	speed_bundled(MB/s)	rates_unbundled	savings
0	gzip 4	3.352383	0.701705	20.010897	2.714575	
1	gzip 5	3.454900	0.710556	17.201050	2.770296	
2	gzip 6	3.491963	0.713628	14.601151	2.792162	
3	gzip 7	3.502496	0.714489	13.574471	2.798391	
4	gzip 8	3.510398	0.715132	11.716914	2.804235	
5	gzip 9	3.510933	0.715175	11.625863	2.804699	
6	brotli 4	3.565088	0.719502	16.478388	2.903769	
7	brotli 5	3.829246	0.738852	11.519678	3.112397	
8	brotli 6	3.863344	0.741157	10.184582	3.128816	
9	brotli 7	3.892121	0.743071	7.572282	3.145081	
10	brotli 8	3.904304	0.743872	5.926585	3.151561	
11	brotli 9	3.917189	0.744715	4.506134	3.159067	
12	brotli 10	4.177070	0.760598	1.226846	3.335079	
13	brotli 11	4.259316	0.765221	0.539299	3.402424	

Group results by non compressed size ranges

In [404]:
```
# ranges are (20000, 100000), (100000, 1000000), (1000000, 3000000)
in bytes
init_sizes_unbundled = np.array(init_sizes_unbundled)
group1 = np.where((init_sizes_unbundled > 2000)*(init_sizes_unbundl
ed <= 100000))[0]
group2 = np.where((init_sizes_unbundled > 100000)*(init_sizes_unbun
dled <= 1000000))[0]
group3 = np.where((init_sizes_unbundled > 1000000)*(init_sizes_unbu
ndled <= 3000000))[0]

print(20000, "-", 100000, "bytes")
frame = pd.DataFrame()
frame["name"] = ["gzip 4", "gzip 5", "gzip 6", "gzip 7", "gzip 8",
"gzip 9",
                 "brotli 4", "brotli 5", "brotli 6", "brotli 7", "b
rotli 8", "brotli 9", "brotli 10", "brotli 11"]


frame["rates_bundled"] = np.hstack((np.mean(np.array(rates_gzip_bun
dled)[group1], axis=0),
                                    np.mean(np.array(rates_brotli_b
undled)[group1], axis=0)))
frame["savings_bundled"] = 1 - 1 / np.hstack((np.mean(np.array(rate
s_gzip_bundled)[group1], axis=0),
                                              np.mean(np.array(rate
s_brotli_bundled)[group1], axis=0)))
frame["speed_bundled(MB/s)"] = np.hstack((np.mean(np.array(speed_gz
ip_bundled)[group1], axis=0),
                                          np.mean(np.array(speed_b
rotli_bundled)[group1], axis=0))) / 1000000

frame["rates_unbundled"] = np.hstack((np.mean(np.array(rates_gzip_u
nbundled)[group1], axis=0),
                                      np.mean(np.array(rates_brotli_u
nbundled)[group1], axis=0)))
frame["savings_unbundled"] = 1 - 1 / np.hstack((np.mean(np.array(ra
tes_gzip_unbundled)[group1], axis=0),
                                                np.mean(np.array(rate
s_brotli_unbundled)[group1], axis=0)))
frame["speed_unbundled(MB/s)"] = np.hstack((np.mean(np.array(speed_
gzip_unbundled)[group1], axis=0),
                                            np.mean(np.array(speed_b
rotli_unbundled)[group1], axis=0))) / 1000000

frame
```

```
20000 - 100000 bytes
```

Out[404]:

	name	rates_bundled	savings_bundled	speed_bundled(MB/s)	rates_unbundled	savings
0	gzip 4	3.219473	0.689390	19.906293	2.639814	
1	gzip 5	3.316049	0.698436	16.875470	2.688858	
2	gzip 6	3.350974	0.701579	14.277811	2.703293	
3	gzip 7	3.360854	0.702457	13.101073	2.708538	
4	gzip 8	3.367366	0.703032	11.325969	2.713746	
5	gzip 9	3.367827	0.703073	11.248899	2.714211	
6	brotli 4	3.414766	0.707154	16.201073	2.794675	
7	brotli 5	3.663572	0.727042	11.228977	2.989190	
8	brotli 6	3.697175	0.729523	9.944989	3.001100	
9	brotli 7	3.725454	0.731576	7.324780	3.012006	
10	brotli 8	3.738544	0.732516	5.745074	3.015555	
11	brotli 9	3.751895	0.733468	4.353647	3.021005	
12	brotli 10	4.005407	0.750338	1.161323	3.181356	
13	brotli 11	4.081090	0.754967	0.511165	3.238609	

In [406]:
```
print(100000, "-", 1000000, "bytes")
frame = pd.DataFrame()
frame["name"] = ["gzip 4", "gzip 5", "gzip 6", "gzip 7", "gzip 8",
"gzip 9",
                 "brotli 4", "brotli 5", "brotli 6", "brotli 7", "b
rotli 8", "brotli 9", "brotli 10", "brotli 11"]
frame["rates_bundled"] = np.hstack((np.mean(np.array(rates_gzip_bun
dled)[group2], axis=0),
                                    np.mean(np.array(rates_brotli_b
undled)[group2], axis=0)))
frame["savings_bundled"] = 1 - 1 / np.hstack((np.mean(np.array(rate
s_gzip_bundled)[group2], axis=0),
                                              np.mean(np.array(rate
s_brotli_bundled)[group2], axis=0)))
frame["speed_bundled(MB/s)"] = np.hstack((np.mean(np.array(speed_gz
ip_bundled)[group2], axis=0),
                                          np.mean(np.array(speed_b
rotli_bundled)[group2], axis=0))) / 1000000

frame["rates_unbundled"] = np.hstack((np.mean(np.array(rates_gzip_u
nbundled)[group2], axis=0),
                                      np.mean(np.array(rates_brotli_u
nbundled)[group2], axis=0)))
frame["savings_unbundled"] = 1 - 1 / np.hstack((np.mean(np.array(ra
tes_gzip_unbundled)[group2], axis=0),
                                                np.mean(np.array(rate
s_brotli_unbundled)[group2], axis=0)))
frame["speed_unbundled(MB/s)"] = np.hstack((np.mean(np.array(speed_
gzip_unbundled)[group2], axis=0),
                                            np.mean(np.array(speed_b
rotli_unbundled)[group2], axis=0))) / 1000000

frame
```

```
100000 - 1000000 bytes
```

Out[406]:

	name	rates_bundled	savings_bundled	speed_bundled(MB/s)	rates_unbundled	savings
0	gzip 4	3.255438	0.692822	21.709783	2.834153	
1	gzip 5	3.352408	0.701707	18.111059	2.901749	
2	gzip 6	3.385023	0.704581	15.608381	2.937161	
3	gzip 7	3.395387	0.705483	14.415784	2.945246	
4	gzip 8	3.402633	0.706110	12.668371	2.952941	
5	gzip 9	3.403091	0.706150	12.492436	2.953505	
6	brotli 4	3.461648	0.711120	17.202166	3.086969	
7	brotli 5	3.710503	0.730495	11.969930	3.321406	
8	brotli 6	3.740765	0.732675	10.554053	3.346479	
9	brotli 7	3.764571	0.734365	7.812185	3.373016	
10	brotli 8	3.774702	0.735078	6.037380	3.384885	
11	brotli 9	3.785685	0.735847	4.591869	3.396362	
12	brotli 10	4.033893	0.752101	1.202672	3.598985	
13	brotli 11	4.104526	0.756367	0.523913	3.683567	

In [407]:
```
print(1000000, "-", 3000000, "bytes")
frame = pd.DataFrame()
frame["name"] = ["gzip 4", "gzip 5", "gzip 6", "gzip 7", "gzip 8",
"gzip 9",
                 "brotli 4", "brotli 5", "brotli 6", "brotli 7", "b
rotli 8", "brotli 9", "brotli 10", "brotli 11"]
frame["rates_bundled"] = np.hstack((np.mean(np.array(rates_gzip_bun
dled)[group3], axis=0),
                                    np.mean(np.array(rates_brotli_b
undled)[group3], axis=0)))
frame["savings_bundled"] = 1 - 1 / np.hstack((np.mean(np.array(rate
s_gzip_bundled)[group3], axis=0),
                                              np.mean(np.array(rate
s_brotli_bundled)[group3], axis=0)))
frame["speed_bundled(MB/s)"] = np.hstack((np.mean(np.array(speed_gz
ip_bundled)[group3], axis=0),
                                          np.mean(np.array(speed_b
rotli_bundled)[group3], axis=0))) / 1000000

frame["rates_unbundled"] = np.hstack((np.mean(np.array(rates_gzip_u
nbundled)[group3], axis=0),
                                      np.mean(np.array(rates_brotli_u
nbundled)[group3], axis=0)))
frame["savings_unbundled"] = 1 - 1 / np.hstack((np.mean(np.array(ra
tes_gzip_unbundled)[group3], axis=0),
                                                np.mean(np.array(rate
s_brotli_unbundled)[group3], axis=0)))
frame["speed_unbundled(MB/s)"] = np.hstack((np.mean(np.array(speed_
gzip_unbundled)[group3], axis=0),
                                            np.mean(np.array(speed_b
rotli_unbundled)[group3], axis=0))) / 1000000

frame
```

```
1000000 - 3000000 bytes
```

Out[407]:

	name	rates_bundled	savings_bundled	speed_bundled(MB/s)	rates_unbundled	savings
0	gzip 4	3.207758	0.688256	22.311194	2.806185	
1	gzip 5	3.298249	0.696809	19.365311	2.861184	
2	gzip 6	3.327129	0.699441	15.542296	2.875304	
3	gzip 7	3.334811	0.700133	13.912655	2.881370	
4	gzip 8	3.338078	0.700426	12.395455	2.884158	
5	gzip 9	3.338087	0.700427	12.474473	2.884215	
6	brotli 4	3.389592	0.704979	18.884871	2.948077	
7	brotli 5	3.631806	0.724655	12.310412	3.142986	
8	brotli 6	3.662058	0.726929	10.677770	3.153725	
9	brotli 7	3.687079	0.728783	7.824110	3.164575	
10	brotli 8	3.696520	0.729475	5.929425	3.167617	
11	brotli 9	3.705561	0.730135	4.381963	3.171627	
12	brotli 10	3.980056	0.748747	1.136612	3.350162	
13	brotli 11	4.047179	0.752914	0.514281	3.417089	

Compare the results for each example

In [389]:
```
ratio_of_rates = []
ratio_of_times = []
for i in range(len(rates_gzip_unbundled)):
    ratio_of_rates.append(np.hstack((np.array(rates_gzip_bundled[i]
) / np.array(rates_gzip_unbundled[i]),
                                     np.array(rates_brotli_bundled[
i]) / np.array(rates_brotli_unbundled[i]))))
    ratio_of_times.append(np.hstack((np.array(times_gzip_bundled[i]
) / np.array(times_gzip_unbundled[i]),
                                     np.array(times_brotli_bundled[
i]) / np.array(times_brotli_unbundled[i]))))

frame = pd.DataFrame()
frame["name"] = ["gzip 4", "gzip 5", "gzip 6", "gzip 7", "gzip 8",
"gzip 9",
                 "brotli 4", "brotli 5", "brotli 6", "brotli 7", "b
rotli 8", "brotli 9", "brotli 10", "brotli 11"]
frame["ratio of rates"] = np.mean(ratio_of_rates, axis=0)
frame["ratio of times"] = np.mean(ratio_of_times, axis=0)
frame
```

Out[389]:

	name	ratio of rates	ratio of times
0	gzip 4	1.427097	0.632208
1	gzip 5	1.453109	0.749165
2	gzip 6	1.464445	0.894621
3	gzip 7	1.467382	0.906732
4	gzip 8	1.469844	1.163363
5	gzip 9	1.470023	1.152780
6	brotli 4	1.393734	0.490474
7	brotli 5	1.409129	0.684028
8	brotli 6	1.418572	0.757256
9	brotli 7	1.426384	1.009216
10	brotli 8	1.429969	1.295686
11	brotli 9	1.433369	1.711266
12	brotli 10	1.460546	5.122315
13	brotli 11	1.465019	9.145809