Kafka 0.10 Compression Benchmark

Posted on 2017-01-03 | 24 Comments

Backgroud

In my previous blog about compression benchmark for Kafka, I have made some tests for Kafka 0.8.2.1. Kafka 0.10 has made a lot of progress, and this post aims to make some benchmaks on Kafka 0.10.

In this post, I'm going to test 3 parts:

1. Producer - time cost, throughput, bandwidth, total traffic
2. Consumer - time cost, throughput
3. Capacity - disk usage, server/client CPU usage

Environment

Hardware Box

I use Docker on Mac to run two containers - zk and kafka.

Mac CPUs	Mac Memory	Mac Disk	Docker CPUs	Docker Memory
2.5 GHz Intel Core i7	16GB	512GB SSD	4	2G

`docker-compose.yml` is below:

```
version: "2"

services:
  zk:
    image: kafka-0.10_zk:1
    ports:
      - "2181:2181"

  kafka:
    depends_on:
      - zk
    image: kafka-0.10_kafka:1
    ports:
      - "9092:9092"
      - "9999:9999"
    command: --override zookeeper.connect=zk:2181
```

Software Box

All Kafka JVM parameters are default because the benchmark's main purpose is to compare different compression algorithm to the `none` compression.

Kafka Version	JDK	Scala	Broker	Kafka Replica	Kafka Partition
0.10.1.0	1.7.0-b147	2.11	1	1	1

Log Files

The log file comes from nginx access logs, and it use 1.2GB disk space and it has 5,190,426 lines.

```
[root@780e74bff80d kafka_2.11-0.10.1.0]# du -sh nginx.log
1.2G    nginx.log
```

```
[root@780e74bff80d kafka_2.11-0.10.1.0]# wc -l nginx.log
5190426 nginx.log
[root@780e74bff80d kafka_2.11-0.10.1.0]# tail -1 nginx.log
127.0.0.1       hello.com       127.0.0.1       127.0.0.1       - 30/Dec/2016:09:43:28 +0800 POST /index.php HTTP/1.1 200 64 -Mozilla/4.0 (co
```

Producer Benchmark

To test Producer performance, I will use `kafka-console-producer.sh` to send `nginx.log` to kafka, `dstat` to get network metrics and `time` in linux to get the time costs. The details are below:

1. time(second) - by Linux command `time`
2. throughtput(message/second) - `total lines` / `time`
3. bandwidth(MB/s) - by `dstat -nT`
4. traffic(MB) - `bandwidth` * `time`

The command to send messsages is:

```
[root@a7dd0e808964 kafka_2.11-0.10.1.0]# time bin/kafka-console-producer.sh --broker-list a7dd0e808964:9092 --topic test_producer --batch-siz
```

The command to get network metrics is:

```
[root@a7dd0e808964 kafka_2.11-0.10.1.0]# dstat -nT
-net/total- --epoch---
 recv  send|  epoch
   0     0 |1483439091
  86B  144B|1483439092
   0     0 |1483439093
```

Detail Metrics for Every Compression Codecs

Compression Codec: none

batch.size	time(sec)	throughput(msg/s)	bandwidth (MB/s)	traffic (MB)
500	35.067	148014.54	38.3	1343.07
1000	35.579	145884.54	37.2	1323.54
1500	35.656	145569.5	38.3	1365.62
5000	31.905	162683.78	41.8	1333.63
10000	35.212	147405.03	38.3	1348.62
AVERAGE	34.68	149911.48	38.78	1342.9

Compression Codec: gzip

batch.size	time(sec)	throughput(msg/s)	bandwidth (MB/s)	traffic (MB)
500	73.302	70808.79	2.0	146.6
1000	68.695	75557.55	2.0	137.39
1500	72.471	71620.73	2.0	144.94
5000	76.469	67876.21	2.0	152.94

batch.size	time(sec)	throughput(msg/s)	bandwidth (MB/s)	traffic (MB)
10000	73.865	70269.09	2.0	147.73
AVERAGE	72.96	71226.47	2.0	145.92

Compression Codec: snappy

batch.size	time(sec)	throughput(msg/s)	bandwidth (MB/s)	traffic (MB)
500	29.163	177979.84	24.5	714.49
1000	33.959	152843.9	21.4	726.72
1500	31.152	166616.14	23.5	732.07
5000	27.420	189293.440	28.0	767.76
10000	28.019	185246.65	26.9	753.71
AVERAGE	29.94	174395.99	24.86	738.95

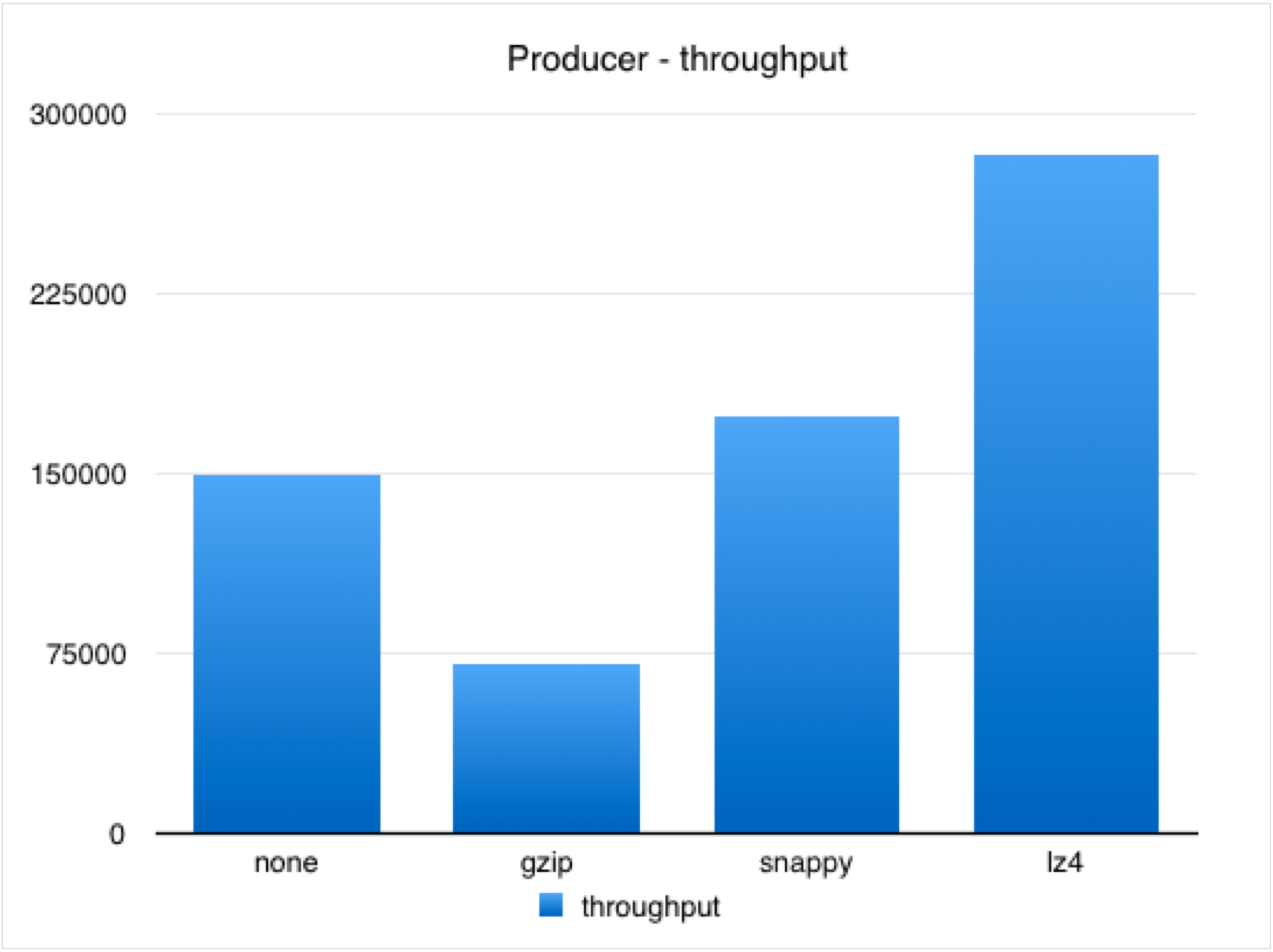
Compression Codec: lz4

batch.size	time(sec)	throughput(msg/s)	bandwidth (MB/s)	traffic (MB)
500	17.937	289369.79	14.01	251.3
1000	17.837	290992.1	13.70	244.37
1500	17.143	302772.33	14.03	240.52
5000	18.525	280184.94	13.43	248.79
10000	20.567	252366.7	11.76	241.87
AVERAGE	18.4	283137.17	13.39	245.37

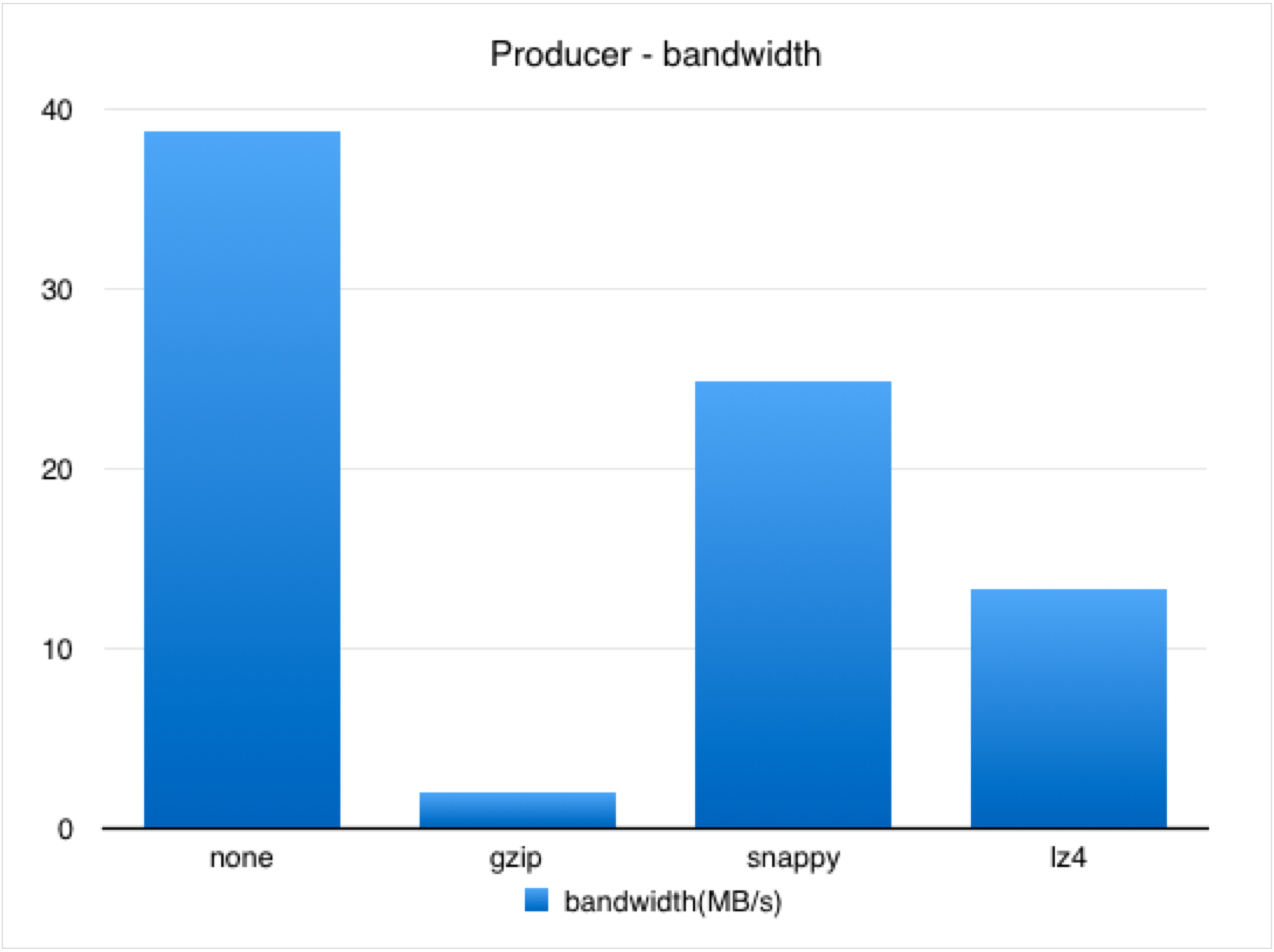
Summary

codec	throughput	bandwidth	traffic
none	149911.48	38.78	1342.9
gzip	71226.47	2.0	145.92
snappy	174395.99	24.86	738.95
lz4	283137.17	13.39	245.37

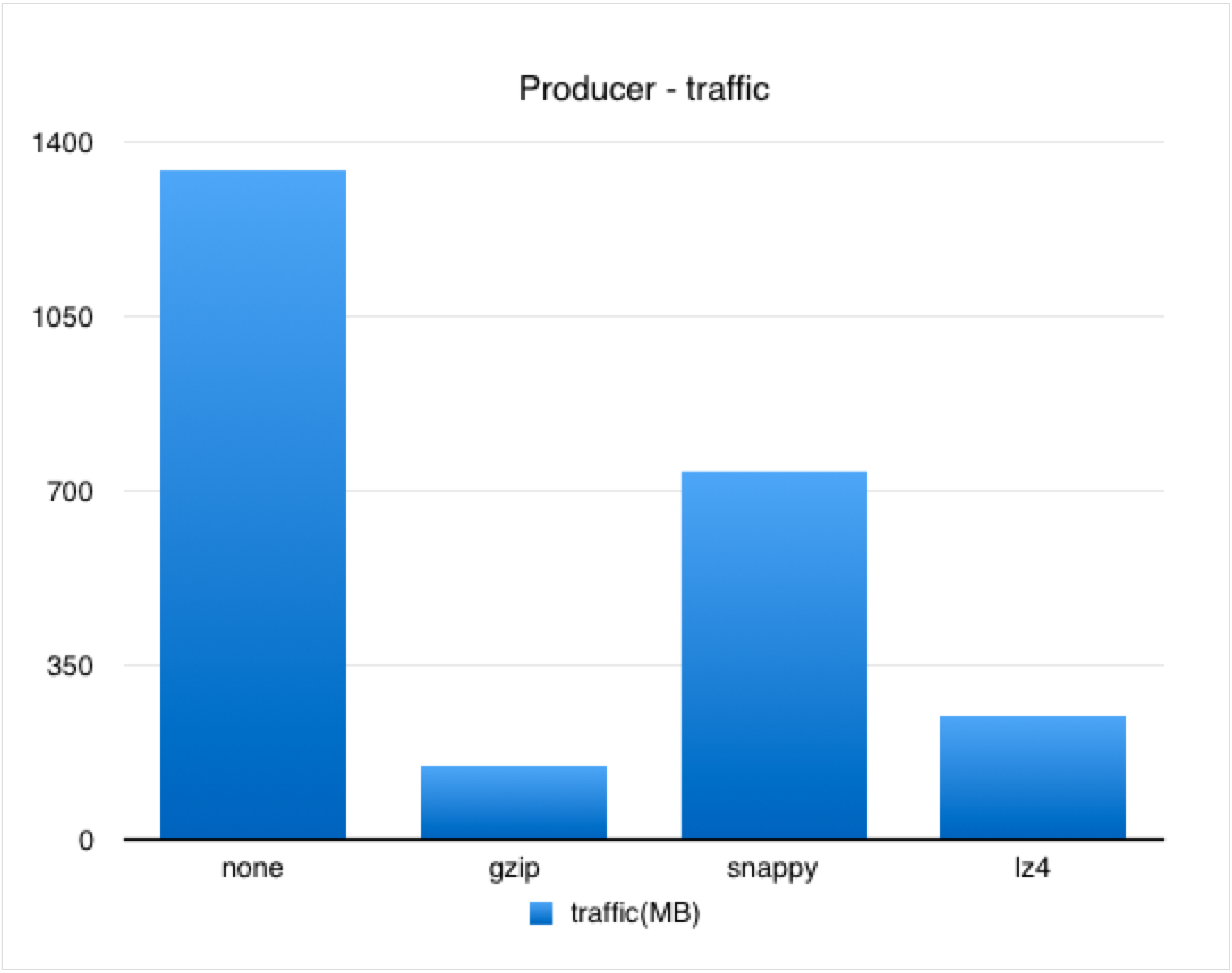
Throughput Overview



Bandwidth Overview

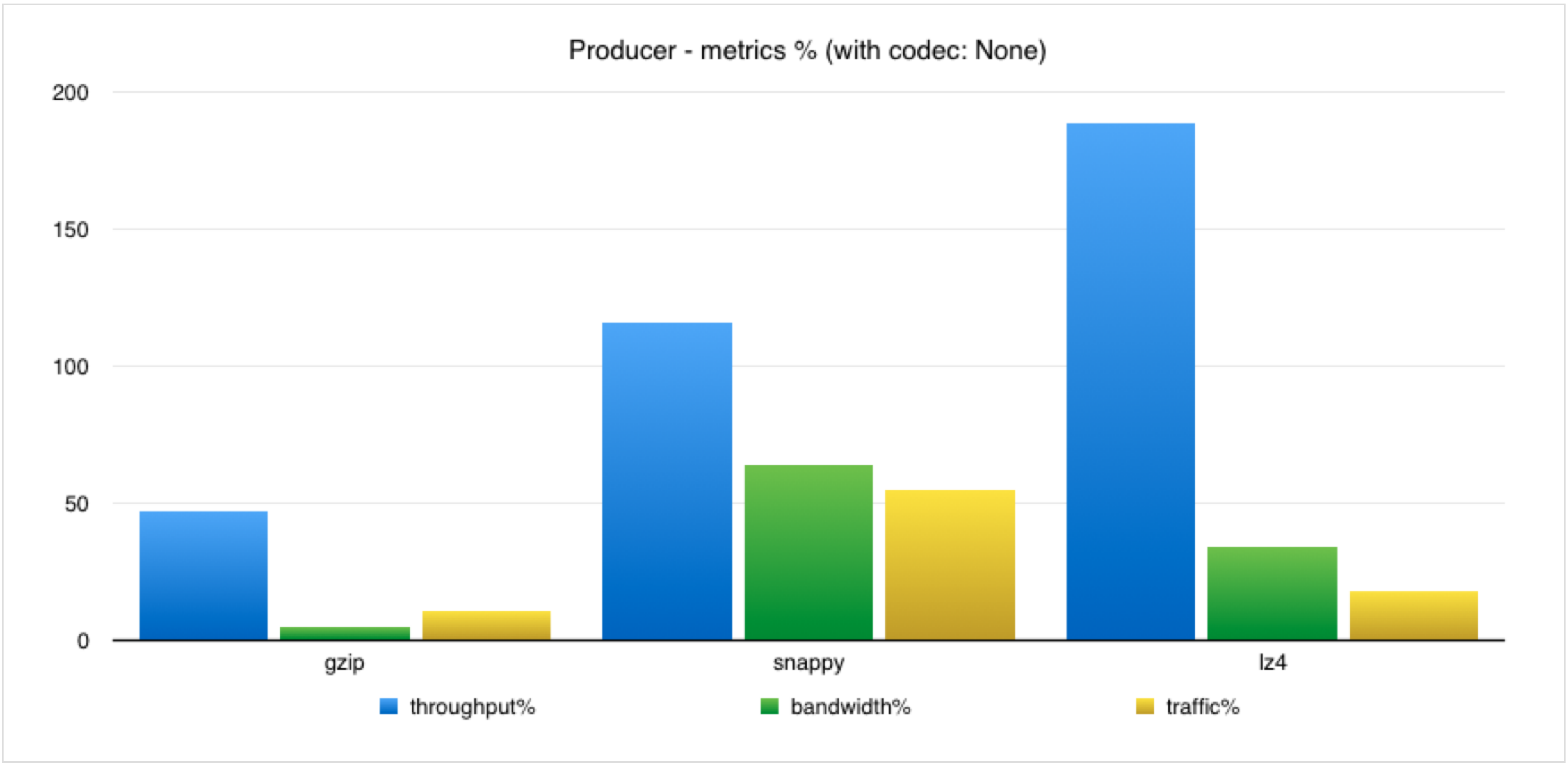


Traffic Overview



Percentage Overview

codec	throughput%	bandwidth%	traffic%
gzip	47.51	5.16	10.87
snappy	116.33	64.11	55.03
lz4	188.87	34.53	18.27



Consumer Benchmark

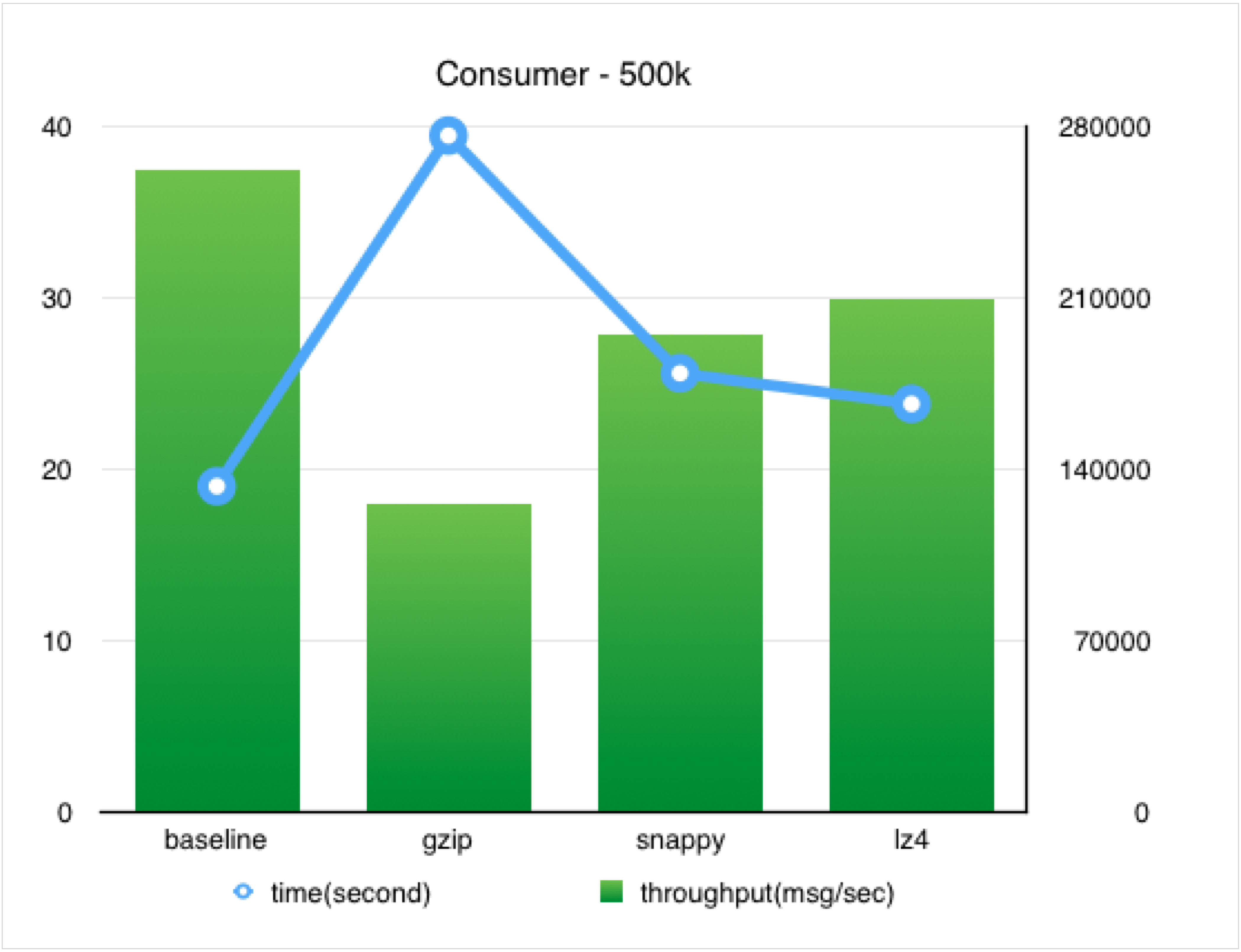
Tests for `Consumer` is much more easier than the one for `Producer`. Before tests, I will send the same `nginx.log` to Kafka with different compression codec - `none`, `gzip`, `snappy` and `lz4`. And use `kafka-console-consumer.sh` to consume a fixed number of messages and in this tests the number is 500k(5,000,000). What I need to look for is the time the procedure costs, and furthermore, we can get the throughput.

The bash command is:

```
time bin/kafka-console-consumer.sh --bootstrap-server 97200db31e2c:9092 --topic consumer_none --max-messages 5000000 --from-beginning > /dev/
```

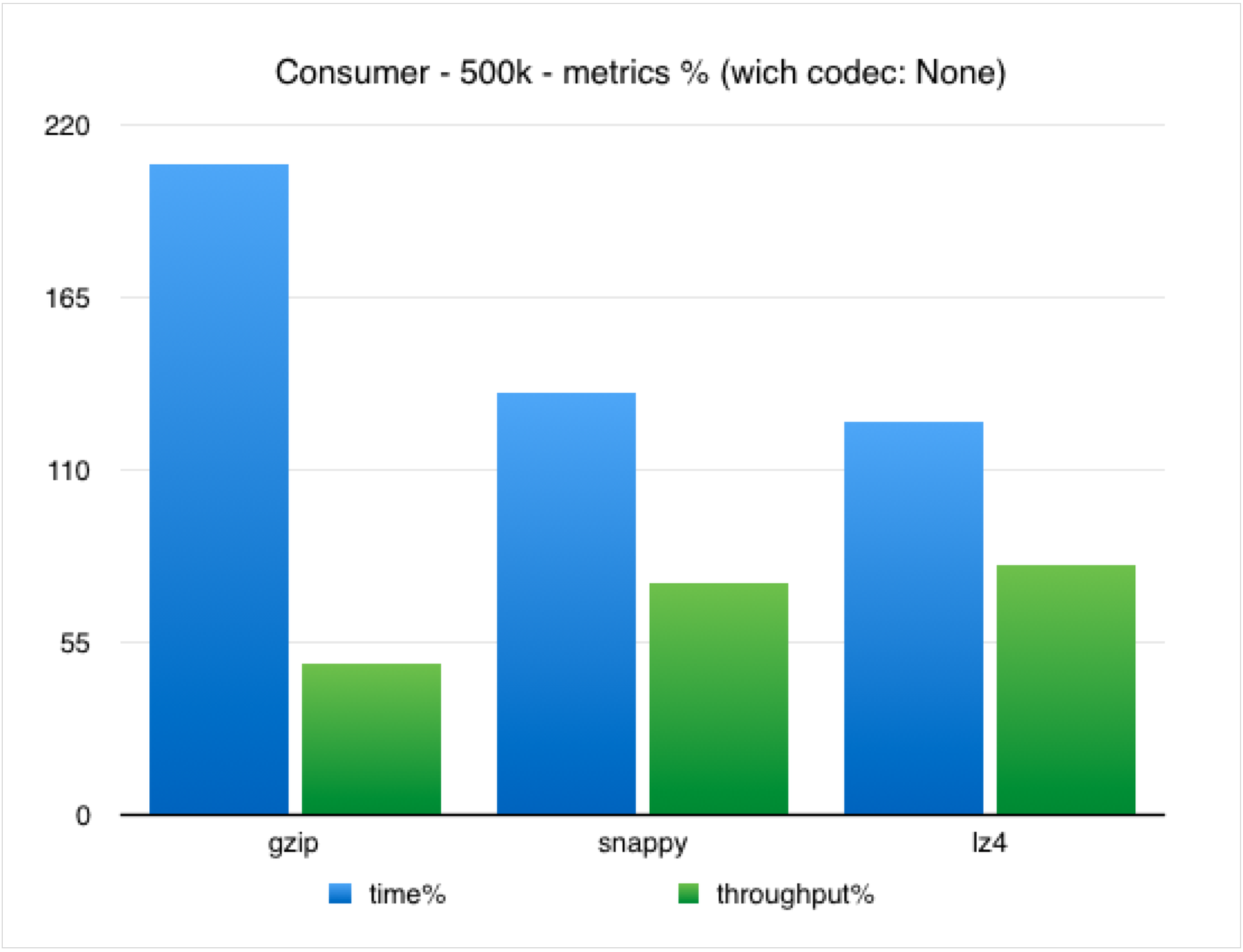
Detail Metrics

codec	time(second)	throughput(msg/s)
none	19.046	262522.31
gzip	39.493	126604.71
snappy	25.632	195068.66
lz4	23.846	209678.77



Percentage:

codec	time%	throughput%
gzip	207.36	48.23
snappy	134.58	74.31
lz4	125.2	79.87



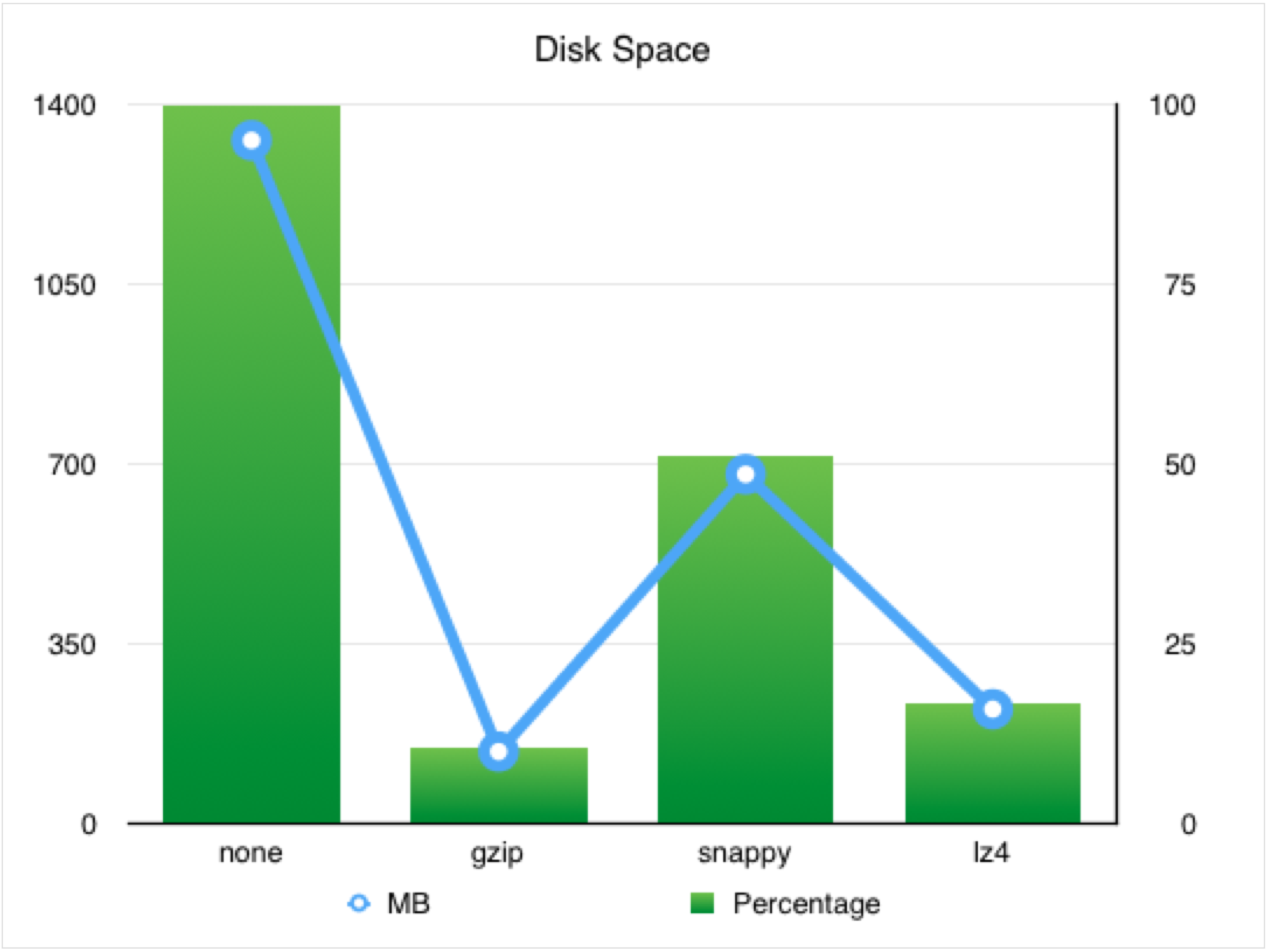
Capacity Benchmark

In the previous blog, I have not make tests for this section. The pressure each codec cause to the CPU is another important factor to consider. I will make some simple benchmark in disk space and CPU in this section by `dstat`.

Disk Usage

Although Kafka has its own retention policies, and it works well, but sometimes the disk space could be in engineers' consideration, especially in large Kafka cluster. In the former section, I have sent `nginx.log` to Kafka with different codecs and I measured the the disk space each topic has used. The numbers can simply be got by `du -sh` in Kafka logs directory.

codec	Disk Space(MB)	Percentage%
none	1329.53	100
gzip	140.18	10.54
snappy	679.81	51.13
lz4	222.58	16.74



CPU Usage

Compression and de-compression will mainly use cpu and I will record the `usr`, `sys`, `wait` and the total of them to measure how much CPU each codec will use. The data is made by `dstat` as well. Pay attention that my docker only has 4 CPU and this tests are mainly used to compare with different codec, not to dig into the absoulute number because it would be different in different boxes.

The test is simple, I used `dstat` to record the system metrics I want, and meanwhile, use `kafka-console-producer.sh` or `kafka-console-consumers.sh` in another container (not the Kafka container) to send or consume data from Kafka.

I will record metrics of both client(run console shell) and server(run Kafka server).

Producer CPU Usage

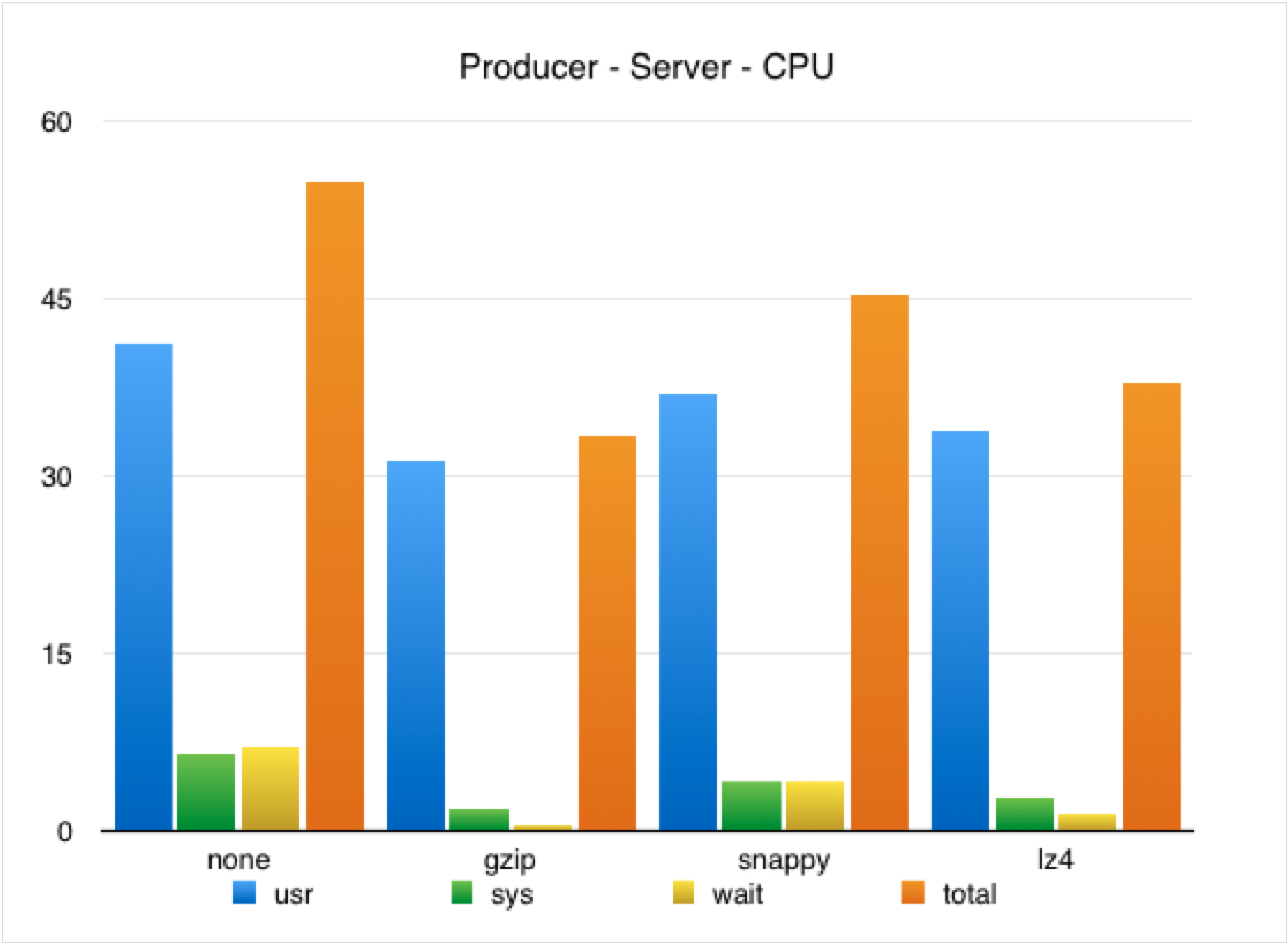
Server Side

Metrics

codec	usr	sys	wait	total
none	41.08	6.56	7.08	54.72
gzip	31.28	1.70	0.45	33.43

codec	usr	sys	wait	total
snappy	36.89	4.12	4.13	45.14
lz4	33.72	2.77	1.37	37.86

Chart

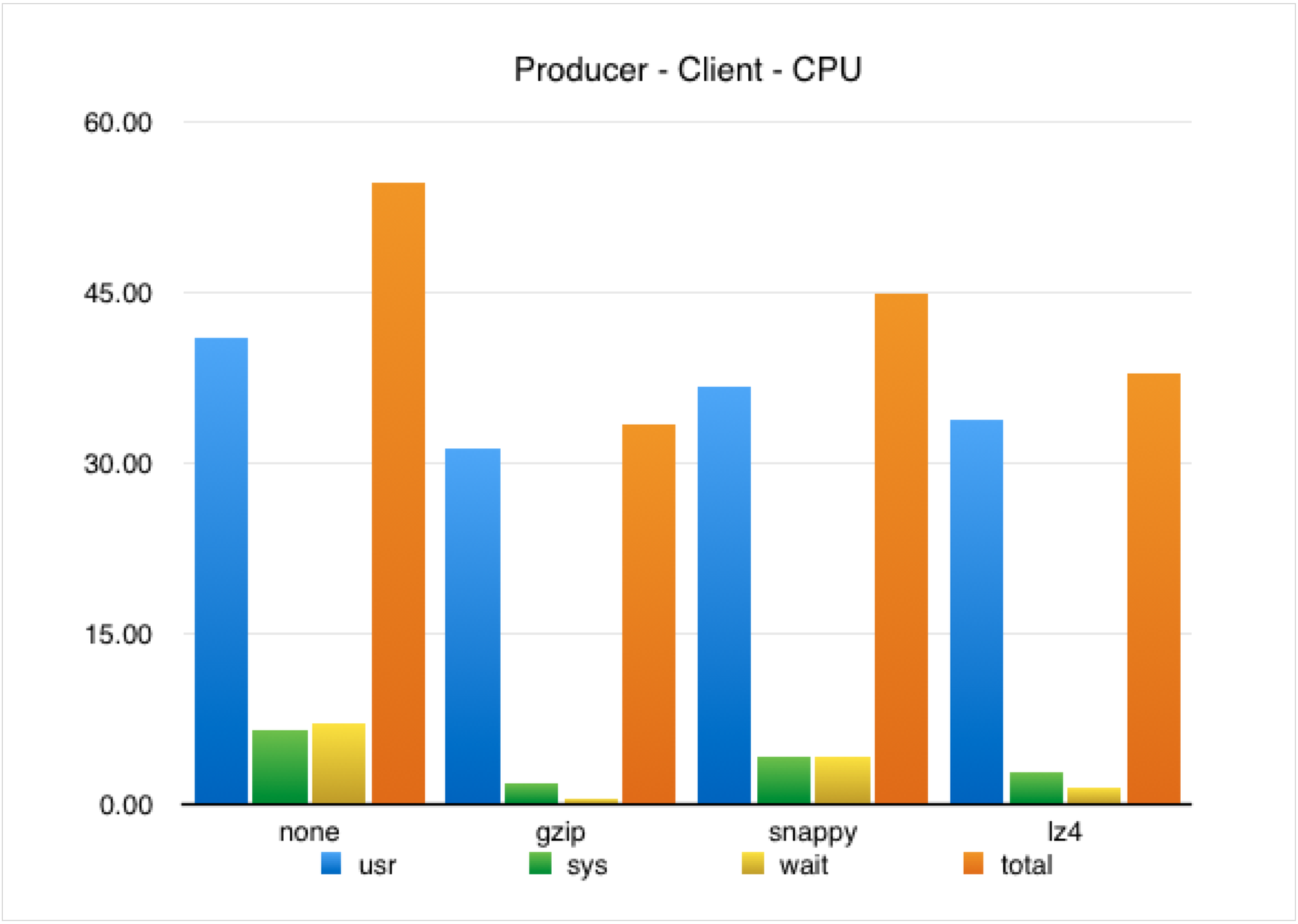


Client side

Metrics

codec	usr	sys	wait	total
none	41.00	6.52	7.09	54.61
gzip	31.28	1.70	0.44	33.42
snappy	36.65	4.13	4.07	44.85
lz4	33.70	2.76	1.36	37.82

Chart



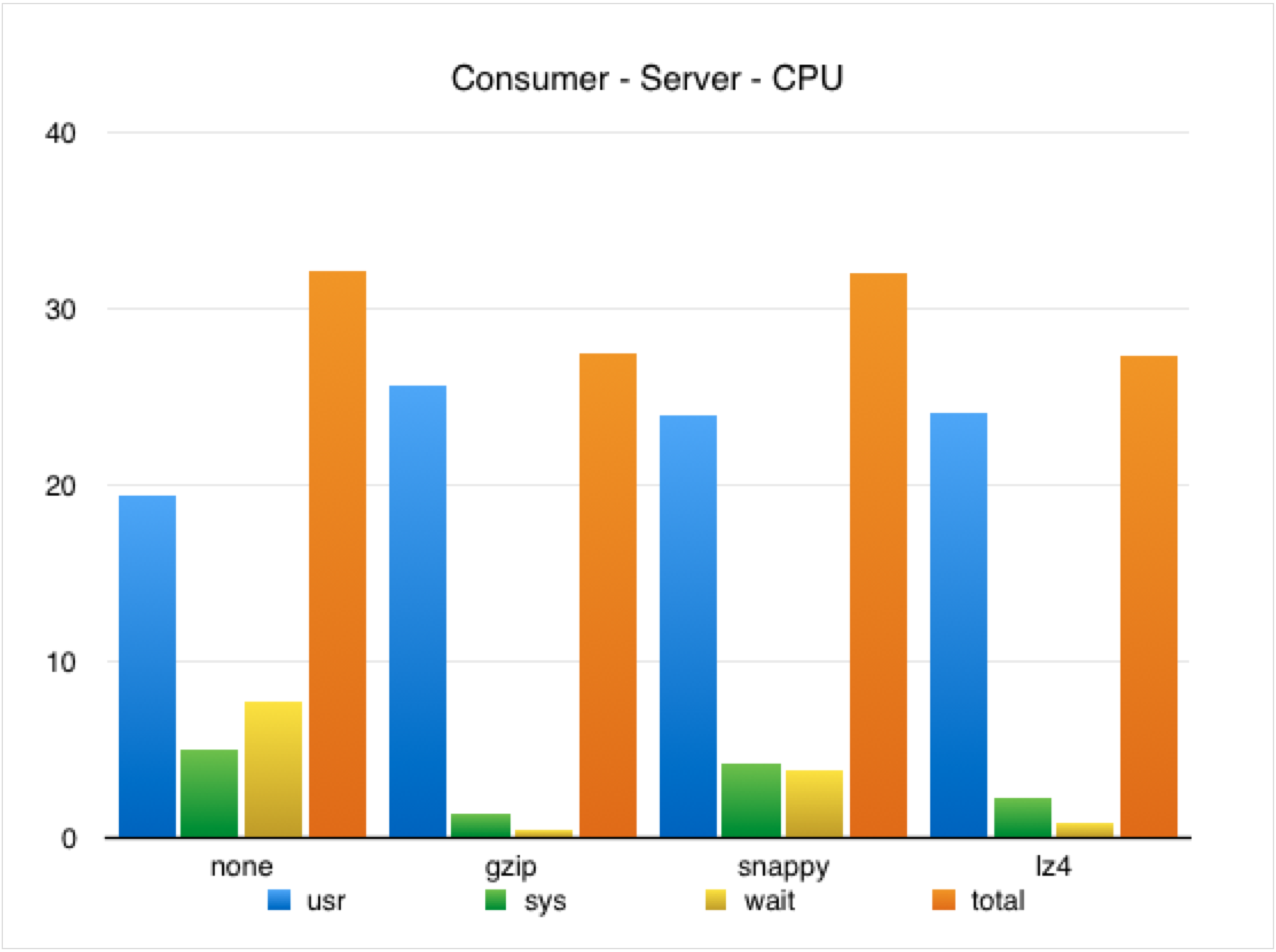
Consumer CPU Usage

Server side

Metrics

codec	usr	sys	wait	total
none	19.41	5.05	7.74	32.2
gzip	25.63	1.43	0.47	27.53
snappy	23.92	4.23	3.90	32.05
lz4	24.13	2.35	0.90	27.38

Chart

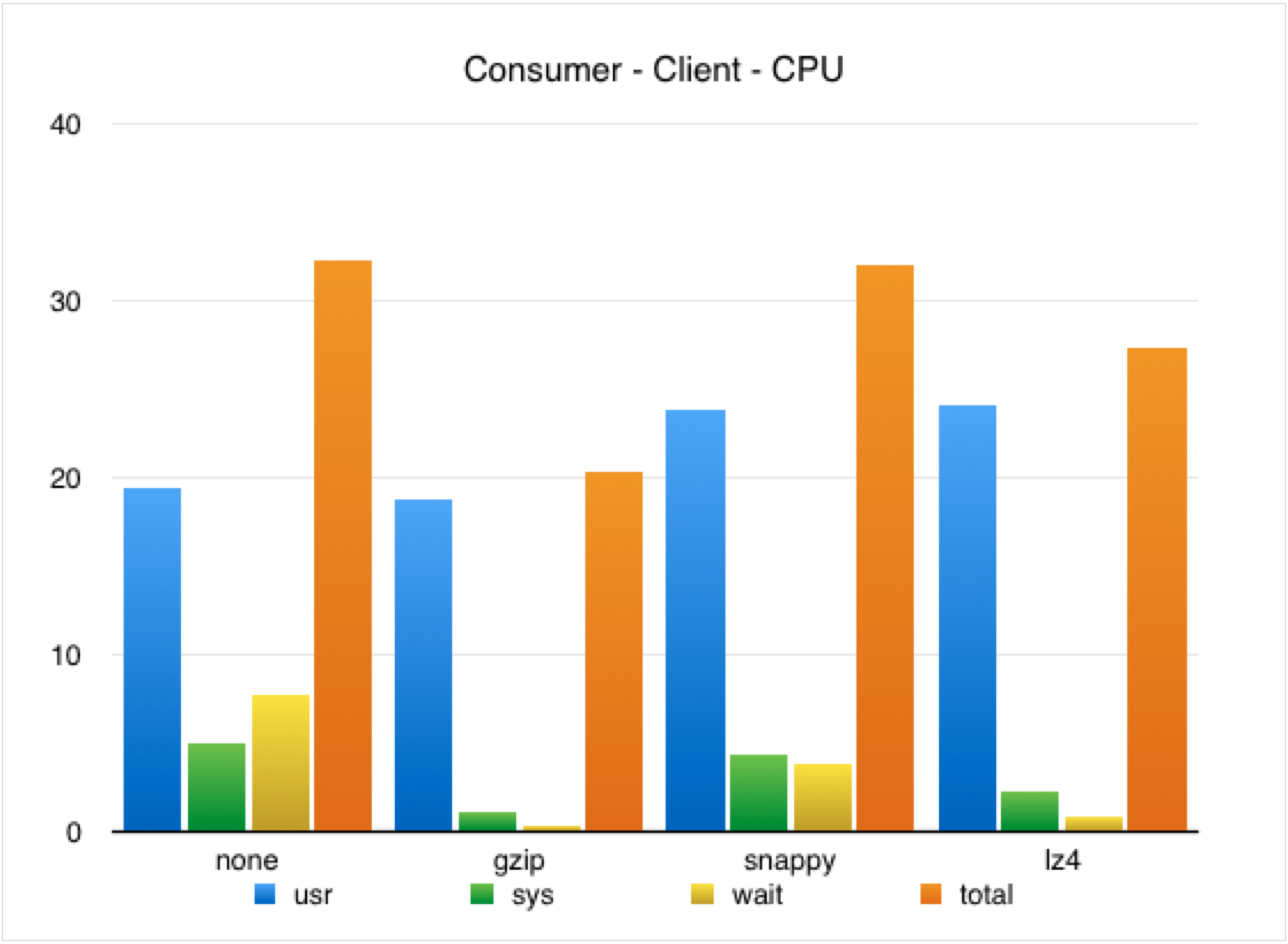


Client Side

Metrics

codec	usr	sys	wait	total
none	19.47	5.05	7.77	32.29
gzip	18.84	1.16	0.41	20.41
snappy	23.82	4.36	3.86	32.04
lz4	24.13	2.35	0.91	27.39

Chart



Conclusion

`GZIP` has the best compression rate but lowest performance, and `LZ4` has the best performance. In the aspect of capacity, `GZIP` and `NONE` will cause wome `wait` which I don't know the reason for that. Actually, the CPU usage for each codec is almost the same, I think capacity won't be the main cause to choose different codecs.

To summarize the benchmarks briefly, use `GZIP` if you need cost less bandwidth and disk space, use `LZ4` to maximum the performance

There is also one problem this benchmark has not cover - how much CPU usage would Kafka server use when there are a huge number of clients. Will the increase of the server-side CPU usage be the linear growth with the number of client? I have not made this test because I only have two containers.

#Kafka

< Join ViSenze in Singapore

The Comparsion Between Apache Flume and Heka >

24 Comments Renjie Yao Blog

1 Login

Recommend 8 Share

Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

trang anh • 2 years ago

Great post, Thanks

1 ^ | ˅ • Reply • Share ›

yaorenjie Mod → trang anh • 2 years ago

Thank you you like it

^ | ˅ • Reply • Share ›

trang anh → yaorenjie • 2 years ago

it is also influnced by message size. I have tested with record size of 100 bytes, and compression happned.

Below as result for 1M message with the size of 100.
64M ./test_none-0
14M ./test_snappy-0

The fact is that the default buffer memory is hardcoded of 1024 bytes so any message size is greater than this value, this leads ignoring the compression for snappy, there was an attempt to fix this one in kafka 10, but it was reverted due to performance and memory issues.

https://issues.apache.org/j...

^ | ˅ • Reply • Share ›

trang anh → yaorenjie • 2 years ago

i am having some confusion about snappy it does NOT does any compression on data when it is write onto disk , and exactly same as NONE. I am not sure why is that?

^ | ˅ • Reply • Share ›

yaorenjie Mod → trang anh • 2 years ago

In my test, `none` costs 1329.53 MB in disk while `snappy` costs 679.81 MB, which is almost half of `none`.

^ | ˅ • Reply • Share ›

trang anh → yaorenjie • 2 years ago

I wrote a wrapper script to automate the test as below:

```
echo "performance test"
CODES=( 'none' 'snappy')
message_size=(100 1000 2000 5000 10000)
KAFKA_HOME=/home/sysgenu/kafka/kafka_2.10-0.10.0.1
CMD=''
NUM_RECORD=$1
counter=0
TOPIC_NAME=''
datetime(){
echo `date +'%Y%m%d_%H%M%S'`
}
for i in ${CODES[@]} ; do
for j in ${message_size[@]} ; do
TOPIC_NAME="test_${i}"
timestamp=`datetime`
echo `date` > "pef_${i}_${counter}_$timestamp.log"
```

see more

^ | ˅ • Reply • Share ›

yaorenjie Mod → trang anh • 2 years ago

I think you actually have not use snappy. A easiest way to check if the snappy make effects, you can tail the .log file in kafka-logs directory, your topic. If the snappy codec is in effect, you will see some messy codes. If you find it is readable chars in file, that is to say your codec is not in use.

^ | ˅ • Reply • Share ›

trang anh → yaorenjie • 2 years ago

I did the same for gzip and lz4 and see the difference, anyway i need to check to see real content as you advised at home. THanks vey much for your support. I will let you know as it is so confusing me

^ | ˅ • Reply • Share ›

yaorenjie Mod → trang anh • 2 years ago

Yes, can you just test by my command written in the article? I also wanna know if there is something wrong. BTW, kafka-perf-test.sh is not very good in my early tests:)

^ | ˅ • Reply • Share ›

trang anh → yaorenjie • 2 years ago

Can you upload the data test to somewhere, then i can work on the same data test to simulate to see any difference?

^ | ˅ • Reply • Share ›

yaorenjie Mod ➦ trang anh • 2 years ago
I've deleted, it is a nginx access log. Or you can give me yours?
^ | ˅ • Reply • Share ›

trang anh ➦ yaorenjie • 2 years ago
Actualy there is an issue with kafka as it uses default buffer block size.
https://issues.apache.org/j...
^ | ˅ • Reply • Share ›

trang anh ➦ yaorenjie • 2 years ago
I used the producer testing test as a part of kafka release. Basically I passed serveral parametters such compression.type. Againt I tried at home but result still the same, I am not sure is there any problem with the latest version. `2.10-0.10.1.1`

Below is the script used to test at home

record_size=$1
codecs=('none' 'snappy')
batch_size=1000
cmd=
kafka_home=/Users/anhtrang/working/softwares/kafka_2.10-0.10.1.1
file_name=
PROC_ID=
topic_name=
for i in ${codecs[@]} ; do
topic_name="test_${i}"
cmd="$kafka_home/bin/kafka-producer-perf-test.sh --num-records 1000000 --record-size=${record_size} --throughput 100000 --topic ${topic_name} --producer-props bootstrap.servers=localhost:9092 compression.type=${i}"
echo $cmd

see more

^ | ˅ • Reply • Share ›

devms • 12 days ago
thank you for your post.
^ | ˅ • Reply • Share ›

SeiQueNadaSei • a month ago
Thank you , sr. Great work!
^ | ˅ • Reply • Share ›

Soumen Mukherjee • 8 months ago
Great Post Yao ... Do you have any plans to update this post perhaps with Kafka 1.0 numbers... i think that will give even a better perspective...
^ | ˅ • Reply • Share ›

yaorenjie Mod ➦ Soumen Mukherjee • 8 months ago
Thanks Soumen. I can not promise I'll do it but that is definitely in my plan:)
^ | ˅ • Reply • Share ›

Connie Yang ➦ yaorenjie • 5 months ago
+1 on running these tests on Kafka 1.1. Great post btw.
^ | ˅ • Reply • Share ›

pei w • a year ago
I'm highly confused about the CPU load, the gzip format got the lowest CPU time, but logically it should be worst one, am I wrong? Or I do ignore something I should know. Thanks.
^ | ˅ • Reply • Share ›

yaorenjie Mod ➦ pei w • a year ago
there has issue for snappy (https://stackoverflow.com/q..., but not sure about gzip. For cpu usage of gzip, snappy and lz4, I cannot found useful benchmark on them. perhaps we can try it.
^ | ˅ • Reply • Share ›

josemi • a year ago
Hi,

Is there any way in kafka 0.10.0.x to enable compression only for specific topics?
Is it difficult to believe that kafka does not implement such a thing since it can be potentially dangerous to enable compression all at once in production environments.
Cheers

Br

Reply • Share ›

yaorenjie Mod josemi • a year ago

Sure you can. In the https://kafka.apache.org/do.... The config you write in broker level is a default config for all topics. And topic level config would overwrite it.

Reply • Share ›

trang anh • 2 years ago

Hi,

Problem with snappy compression in the latest release here.
http://stackoverflow.com/qu...

Reply • Share ›

yaorenjie Mod trang anh • 2 years ago

I got it, it may caused by the content pattern and the block size I think.

Reply • Share ›

ALSO ON **RENJIE YAO BLOG**

Kafka Compression Performance Tests

Enable CORS in bottle.py