RateAnalysis

October 23, 2019

1 Introduction

Validating EBDC online compression throughput using the Supermicro SuperWorkstation 7049GP-TRT, with 2 x Intel Xeon Silver 4216 Processor 16-Core 2.1GHz 32 core CPUs and 128 GB memory.

The data is all 2019 sPHENIX TPC SAMPA data at FTBF total 1+TB. The data are buffered on ASUS Hyper M.2 X16 PCIe 3.0 X4 Expansion Card V2 with four SAMSUNG 970 EVO PLUS M.2 2280 1TB PCIe Gen 3.0 x4 NVMe 1.3 V-NAND configured in 4-strip software RAID0. The RAID is tested to 6GBps write and 11GBps write through its PCIe Gen3 x16 interface, matching a large fraction of the FELIX throughput and suppass the expected average rate in sPHENIX year-5 operation.

The data is readout as `parallel` jobs via start-compression.sh, and sink via either `/dev/null` or TPC connections to multiple `ncat` processes either at `localhost` or remote which can be started with start-sink.sh

2 Inputs

[1]:
```
# DataDir = './data_tmp/'
# DataDir = './data_25x_localhost/'
# DataDir = './data_48x_localhost/'
# studytitle = r"$\bf{EBDC}$" + " compression\nlocalhost loopback"
# DataDir = './data_48x_null/'
# studytitle = r"$\bf{EBDC}$" + " compression\noutput to /dev/null"



DataDir = './data_48x_20GbpsNetwork/'
studytitle = r"$\bf{EBDC}$" + " compression\n2x10Gbps ethernet"
# DataDir = './data_60x_20GbpsNetwork/'
# studytitle = r"$\bf{EBDC}$" + " compression\n2x10Gbps ethernet"
# DataDir = './data_25x_20GbpsNetwork/'
# studytitle = r"$\bf{EBDC}$" + " compression\n2x10Gbps ethernet"
```

[2]:
```
# %matplotlib widget
# %matplotlib ipympl
%matplotlib inline
```

```
# well the html export like dump formats
```

3 Processing

[3]:
```
import os
import ntpath
import re
import pandas as pd
import numpy as np


def processDataset(dataset: str):
    split = dataset.split('-')
    if (len(split) != 3):
        print('skip {}'.format (dataset) );
        return;

    zipcmd = split[0];
    ziplevel = int(split[1]);
    jobs = int(split[2]);
    datasetDir = os.path.join(DataDir, dataset)

    print('processing {}, {} level{} x{}'.format (datasetDir,␣
 ↪zipcmd,ziplevel,jobs) );

    datasubfolders = [os.path.basename(f.path) for f in os.scandir(datasetDir)␣
 ↪if f.is_file() ]
    datasubfolders.sort()
    rpv_in = re.compile('pv_in_([0-9]*)\.log')

    for data in datasubfolders:
#         print ('data = {}'.format(data));
        m = rpv_in.search(data)
        if m is not None:
#             print ('found {} -> {}'.format(data, m.group(1)));
            jobID = m.group(1);
            with open(os.path.join(datasetDir, 'pv_in_{}.log'.format(jobID)))␣
 ↪as f:
                split = f.readlines()[-1].split();
                assert(len(split)==2)
                inTime = float(split[0])
                inSize = float(split[1])
            with open(os.path.join(datasetDir, 'pv_out_{}.log'.format(jobID)))␣
 ↪as f:
                split = f.readlines()[-1].split();
                assert(len(split)==2)
                outTime = float(split[0])
                outSize = float(split[1])

#             print ('df.append {} . {} , {} -> {}'.format(data,␣
 ↪jobID,inSize,outSize));
            dictData = { 'dataset' : dataset ,
                      'zipcmd': zipcmd,
                      'ziplevel': ziplevel,
                      'jobs': jobs,
                      'jobID': jobID,
                      'inTime': inTime,
                      'inSize': inSize,
                      'outTime': outTime  ,
                      'outSize': outSize
                      }
            global dataframe
            dataframe = dataframe.append(dictData, ignore_index=True)



dataframe = pd.DataFrame(columns=['dataset', 'zipcmd', 'ziplevel', 'jobs',␣
 ↪'jobID', 'inTime', 'inSize', 'outTime', 'outSize'])
subfolders = [f.path for f in os.scandir(DataDir) if f.is_dir() ]
subfolders.sort()

for dataset in subfolders:
    processDataset(os.path.basename(dataset))
```

```
skip .ipynb_checkpoints
processing ./data_48x_20GbpsNetwork/gzip-1-48, gzip level1 x48
processing ./data_48x_20GbpsNetwork/gzip-2-48, gzip level2 x48
processing ./data_48x_20GbpsNetwork/gzip-3-48, gzip level3 x48
processing ./data_48x_20GbpsNetwork/gzip-5-48, gzip level5 x48
processing ./data_48x_20GbpsNetwork/gzip-7-48, gzip level7 x48
processing ./data_48x_20GbpsNetwork/gzip-9-48, gzip level9 x48
processing ./data_48x_20GbpsNetwork/lz4-1-48, lz4 level1 x48
processing ./data_48x_20GbpsNetwork/lz4-2-48, lz4 level2 x48
processing ./data_48x_20GbpsNetwork/lz4-3-48, lz4 level3 x48
processing ./data_48x_20GbpsNetwork/lz4-5-48, lz4 level5 x48
processing ./data_48x_20GbpsNetwork/lz4-7-48, lz4 level7 x48
processing ./data_48x_20GbpsNetwork/lz4-9-48, lz4 level9 x48
processing ./data_48x_20GbpsNetwork/lzop-1-48, lzop level1 x48
processing ./data_48x_20GbpsNetwork/lzop-2-48, lzop level2 x48
processing ./data_48x_20GbpsNetwork/lzop-3-48, lzop level3 x48
processing ./data_48x_20GbpsNetwork/lzop-5-48, lzop level5 x48
processing ./data_48x_20GbpsNetwork/lzop-7-48, lzop level7 x48
processing ./data_48x_20GbpsNetwork/lzop-9-48, lzop level9 x48
```

4 Plot

[4]:
```
dataframeSum = pd.DataFrame(columns=['dataset', 'zipcmd', 'ziplevel',␣
↪'jobs','totalInTime', 'totalInSize', 'totalOutTime', 'totalOutSize',␣
↪'Compression', 'inRateGbps', 'outRateGbps'])

zipcmds = dataframe.zipcmd.unique()

for zipcmd in zipcmds:

    zipRows = dataframe.loc[dataframe['zipcmd'] == zipcmd]

    ziplevels = zipRows.ziplevel.unique()
    for ziplevel in ziplevels:
        ziplevelRows = zipRows.loc[zipRows['ziplevel'] == ziplevel]
        print ('processing ', zipcmd, '.',ziplevel, ' size= ',ziplevelRows.
↪size, 'compression ratio = ',ziplevelRows['outSize'].sum()/
↪ziplevelRows['inSize'].sum())
        assert(ziplevelRows.size>1000)

        dictData = { 'dataset' : ziplevelRows['dataset'].iloc[0] ,
                  'zipcmd':  ziplevelRows['zipcmd'].iloc[0] ,
                  'ziplevel':  ziplevelRows['ziplevel'].iloc[0] ,
                  'jobs':  ziplevelRows['jobs'].iloc[0] ,
                    'totalInTime' : ziplevelRows['inTime'].sum() ,
                  'totalInSize':  ziplevelRows['inSize'].sum() ,
                  'totalOutTime':  ziplevelRows['outTime'].sum() ,
                  'totalOutSize':  ziplevelRows['outSize'].sum() ,

                  }

        dictData['Compression'] = dictData['totalOutSize']/␣
↪dictData['totalInSize']
        dictData['inRateGbps'] = dictData['totalInSize']/␣
↪dictData['totalInTime'] * dictData['jobs'] *8/1e9
        dictData['outRateGbps'] = dictData['totalOutSize']/␣
↪dictData['totalOutTime']* dictData['jobs'] *8/1e9

        dataframeSum = dataframeSum.append(dictData, ignore_index=True)
```

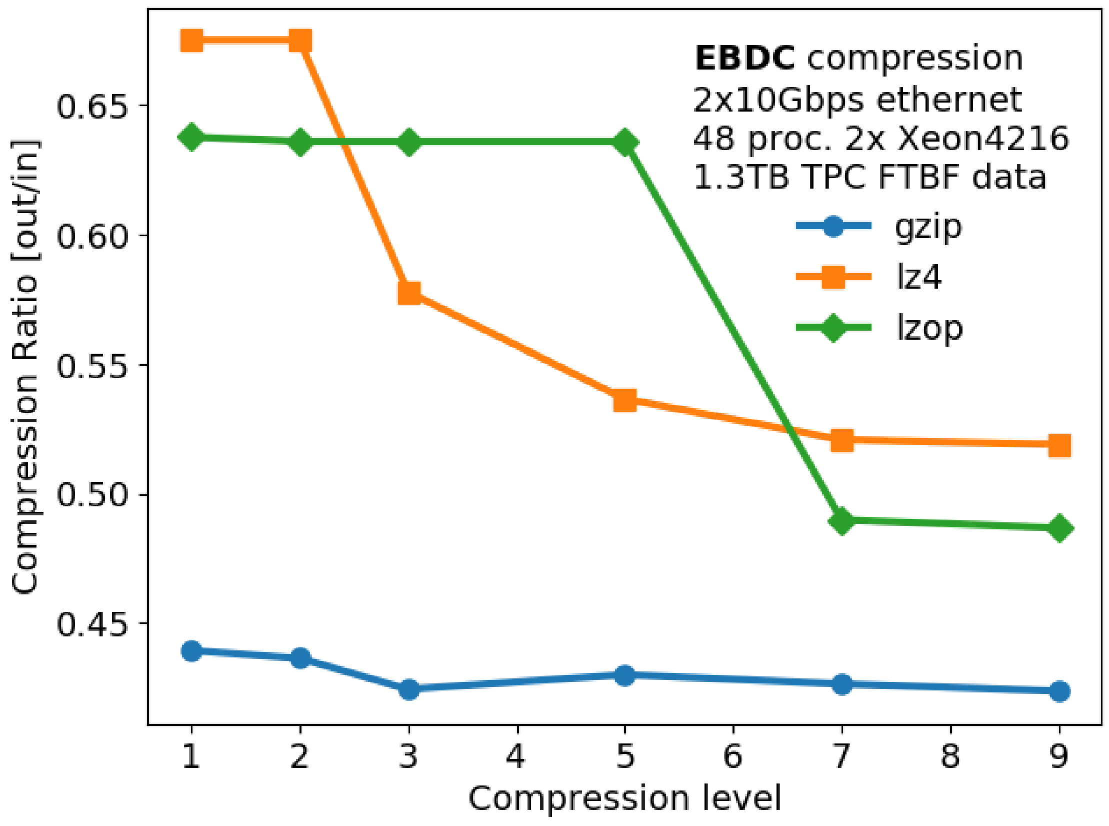
```
processing  gzip . 1  size=  2133 compression ratio =  0.43932139377897234
processing  gzip . 2  size=  2133 compression ratio =  0.43639353842830403
processing  gzip . 3  size=  2133 compression ratio =  0.424364599250152
processing  gzip . 5  size=  2133 compression ratio =  0.4300136029630121
processing  gzip . 7  size=  2133 compression ratio =  0.4264271430634125
processing  gzip . 9  size=  2133 compression ratio =  0.42370792929411943
processing  lz4 . 1  size=  2133 compression ratio =  0.6751259046982664
processing  lz4 . 2  size=  2133 compression ratio =  0.6751259046982664
processing  lz4 . 3  size=  2133 compression ratio =  0.5778558660340661
processing  lz4 . 5  size=  2133 compression ratio =  0.5365085644373812
processing  lz4 . 7  size=  2133 compression ratio =  0.5207546008082999
processing  lz4 . 9  size=  2133 compression ratio =  0.5189943751016245
processing  lzop . 1  size=  2133 compression ratio =  0.6377294848760965
processing  lzop . 2  size=  2133 compression ratio =  0.6359905949774498
processing  lzop . 3  size=  2133 compression ratio =  0.6359905949774498
processing  lzop . 5  size=  2133 compression ratio =  0.6359905949774498
processing  lzop . 7  size=  2133 compression ratio =  0.48989600087271923
processing  lzop . 9  size=  2133 compression ratio =  0.4866962781452209
```

[5]:
```
import matplotlib.pyplot as plt
import numpy as np

Colors = ['#1f77b4',
          '#ff7f0e',
          '#2ca02c',
          '#d62728',
          '#9467bd',
          '#8c564b',
          '#e377c2',
          '#7f7f7f',
          '#bcbd22',
          '#17becf',
          '#1a55FF']
Markers = ['o' , 's',  'D','p' , 'P']

font = {'size'   : 14}
plt.rcdefaults()
plt.rc('font', **font)

studytitle_sup = studytitle + "\n{:d} proc. 2x Xeon4216\n{:.1f}TB TPC FTBF␣
 ↪data".format(
    dataframeSum['jobs'].iloc[0], dataframeSum['totalInSize'].iloc[0]/1e12)
```

4.1 Compression plot

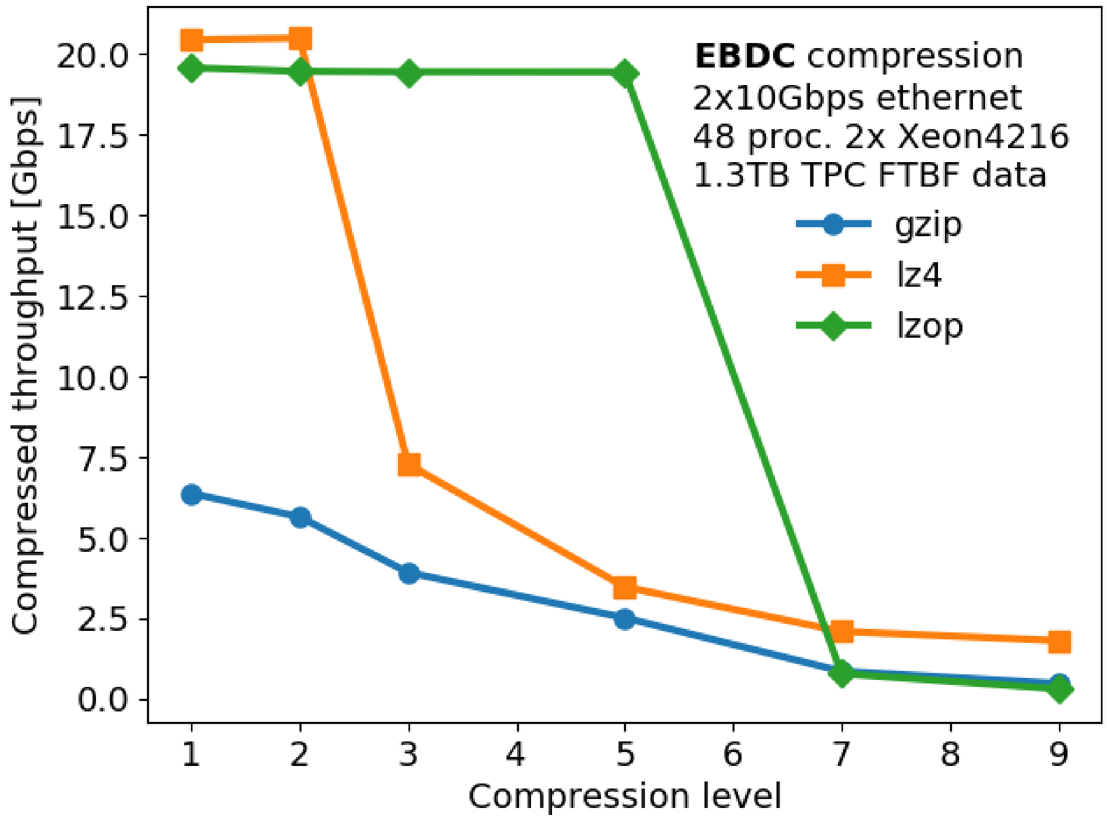
[6]:
```
# dataframeSum.plot(x = 'ziplevel', y = "Compression")

fig = plt.figure()
ax = fig.add_axes([0.15, 0.15, 0.85, 0.85])
plt.xlabel('Compression level')
plt.ylabel('Compression Ratio [out/in]')
```

```
markiter = iter(Markers);
coleriter = iter(Colors);

for zipcmd in dataframeSum.zipcmd.unique():
    zipRows = dataframeSum.loc[dataframeSum['zipcmd'] == zipcmd]
    ax.plot(zipRows['ziplevel'].to_numpy(), zipRows['Compression'].to_numpy(),
            marker=next(markiter), color=next(coleriter), markersize = 8,␣
 ↪linewidth = 3,
            label=zipcmd)

plt.legend(loc='best',title = studytitle_sup, frameon=False)

plt.savefig(os.path.join(DataDir,"Compression.png"), dpi=150)
plt.savefig(os.path.join(DataDir,"Compression.pdf"), dpi=150)
```



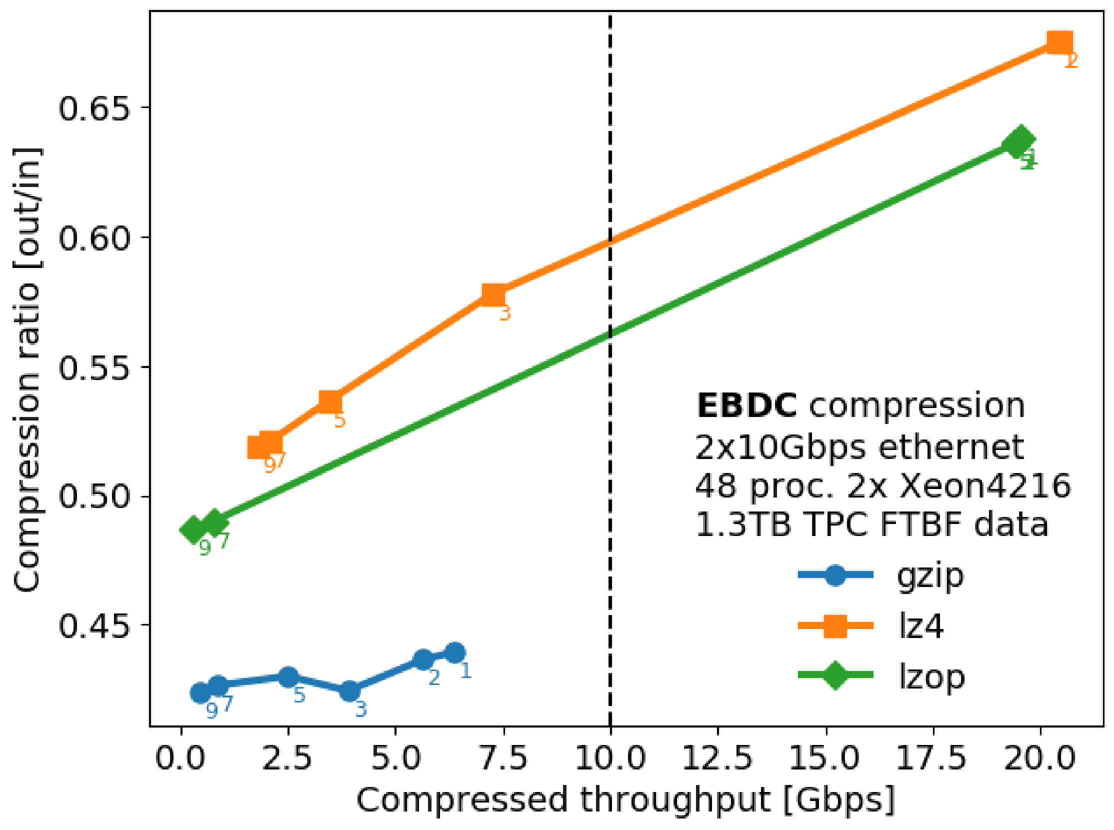
4.2 Compressed throughput

[7]:
```
# dataframeSum.plot(x = 'ziplevel', y = "Compression")

fig = plt.figure()
ax = fig.add_axes([0.15, 0.15, 0.85, 0.85])
plt.xlabel('Compression level')
plt.ylabel('Compressed throughput [Gbps]')

markiter = iter(Markers);
coleriter = iter(Colors);

for zipcmd in dataframeSum.zipcmd.unique():
    zipRows = dataframeSum.loc[dataframeSum['zipcmd'] == zipcmd]
    ax.plot(zipRows['ziplevel'].to_numpy(), zipRows['outRateGbps'].to_numpy(),
            marker=next(markiter), color=next(coleriter), markersize = 8,␣
 ↪linewidth = 3,
            label=zipcmd)

plt.legend(loc='best',title = studytitle_sup, frameon=False)

plt.savefig(os.path.join(DataDir,"Throughput.png"), dpi=150)
plt.savefig(os.path.join(DataDir,"Throughput.pdf"), dpi=150)
```



4.3 Work point curve

[8]:
```
# dataframeSum.plot(x = 'ziplevel', y = "Compression")

fig = plt.figure()
ax = fig.add_axes([0.15, 0.15, 0.85, 0.85])
plt.ylabel('Compression ratio [out/in]')
plt.xlabel('Compressed throughput [Gbps]')

markiter = iter(Markers);
coleriter = iter(Colors);

for zipcmd in dataframeSum.zipcmd.unique():
    zipRows = dataframeSum.loc[dataframeSum['zipcmd'] == zipcmd]
    outRateGbps = zipRows['outRateGbps'].to_numpy()
    Compression = zipRows['Compression'].to_numpy()
    ziplevel = zipRows['ziplevel'].to_numpy()
    c = next(coleriter)
    ax.plot(outRateGbps, Compression,
            marker=next(markiter), color=c, markersize = 8, linewidth = 3,
            label=zipcmd)

    for i in range(0, len(outRateGbps)):
        plt.text(outRateGbps[i]+.1, Compression[i]-.01,str(ziplevel[i]),␣
 ↪fontsize=9, color=c)

#              [str(i) for i in zipRows['ziplevel'].to_numpy()], fontsize=9)

ax.add_line(plt.Line2D([10, 10],ax.get_ylim(), color = 'black', linestyle =␣
 ↪'--'))

plt.legend(loc='best',title = studytitle_sup,
           edgecolor = 'white', frameon=True, facecolor='white', framealpha=0.9)

plt.savefig(os.path.join(DataDir,"FOM.png"), dpi=150)
plt.savefig(os.path.join(DataDir,"FOM.pdf"), dpi=150)
```



5 Scratch

[9]:
```
# %save_html os.path.join(DataDir,"analysis.html")
import sys
from subprocess import check_call

d, fname = os.path.split(sys.executable)
# print (d,fname)
check_call([os.path.join(d,'ipython'), 'nbconvert',
            '--to', 'html',
            'RateAnalysis.ipynb',
            '--output',
            os.path.join(DataDir,"analysis.html")])
check_call([os.path.join(d,'ipython'), 'nbconvert',
            '--to', 'pdf',
            'RateAnalysis.ipynb',
            '--output',
            os.path.join(DataDir,"analysis.pdf")])
```

[9]:
```
0
```

[10]:
```
dataframeSum
```

[10]:
```
       dataset zipcmd  ziplevel  jobs   totalInTime   totalInSize  totalOutTime  \
0    gzip-1-48   gzip         1    48   35681.4667  1.345599e+12    35682.1754
1    gzip-2-48   gzip         2    48   40012.4084  1.345599e+12    40013.1186
2    gzip-3-48   gzip         3    48   56121.1630  1.345599e+12    56122.0689
3    gzip-5-48   gzip         5    48   88668.8855  1.345599e+12    88670.1592
4    gzip-7-48   gzip         7    48  260473.2410  1.345599e+12   260476.5507
5    gzip-9-48   gzip         9    48  469253.2066  1.345599e+12   469258.8453
6     lz4-1-48    lz4         1    48   17078.0360  1.345599e+12    17082.7751
7     lz4-2-48    lz4         2    48   17037.3998  1.345599e+12    17041.7892
8     lz4-3-48    lz4         3    48   41074.9621  1.345599e+12    41087.9044
9     lz4-5-48    lz4         5    48   80064.1867  1.345599e+12    80089.0275
10    lz4-7-48    lz4         7    48  129191.7374  1.345599e+12   129232.2927
11    lz4-9-48    lz4         9    48  149248.8691  1.345599e+12   149296.7664
12   lzop-1-48   lzop         1    48   16856.7238  1.345599e+12    16857.4112
13   lzop-2-48   lzop         2    48   16902.8734  1.345599e+12    16903.5811
14   lzop-3-48   lzop         3    48   16919.5788  1.345599e+12    16920.2196
15   lzop-5-48   lzop         5    48   16922.7959  1.345599e+12    16923.4848
16   lzop-7-48   lzop         7    48  328273.5071  1.345599e+12   328282.7795
17   lzop-9-48   lzop         9    48  841328.1659  1.345599e+12   841350.8210

    totalOutSize  Compression  inRateGbps  outRateGbps
0   5.911506e+11     0.439321   14.481192     6.361771
1   5.872109e+11     0.436394   12.913748     5.635376
2   5.710247e+11     0.424365    9.207047     3.907082
3   5.786260e+11     0.430014    5.827412     2.505831
4   5.738001e+11     0.426427    1.983736     0.845908
5   5.701411e+11     0.423708    1.101133     0.466553
6   9.084490e+11     0.675126   30.255830    20.420828
7   9.084490e+11     0.675126   30.327994    20.469941
8   7.775625e+11     0.577856   12.579687     7.266956
9   7.219256e+11     0.536509    6.453699     3.461391
10  7.007271e+11     0.520755    3.999560     2.082136
11  6.983585e+11     0.518994    3.462071     1.796219
12  8.581284e+11     0.637729   30.653060    19.547563
13  8.557885e+11     0.635991   30.569368    19.441017
14  8.557885e+11     0.635991   30.539186    19.421899
15  8.557885e+11     0.635991   30.533380    19.418152
16  6.592038e+11     0.489896    1.574023     0.771086
17  6.548982e+11     0.486696    0.614160     0.298901
```

[ ]: