RateAnalysis

September 16, 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 TPC connection 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"
```

[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_25x_localhost/gzip-1-25, gzip level1 x25
processing ./data_25x_localhost/gzip-2-25, gzip level2 x25
processing ./data_25x_localhost/gzip-3-25, gzip level3 x25
processing ./data_25x_localhost/gzip-5-25, gzip level5 x25
processing ./data_25x_localhost/gzip-7-25, gzip level7 x25
processing ./data_25x_localhost/gzip-9-25, gzip level9 x25
processing ./data_25x_localhost/lz4-1-25, lz4 level1 x25
processing ./data_25x_localhost/lz4-2-25, lz4 level2 x25
processing ./data_25x_localhost/lz4-3-25, lz4 level3 x25
processing ./data_25x_localhost/lz4-5-25, lz4 level5 x25
processing ./data_25x_localhost/lz4-7-25, lz4 level7 x25
processing ./data_25x_localhost/lz4-9-25, lz4 level9 x25
processing ./data_25x_localhost/lzop-1-25, lzop level1 x25
processing ./data_25x_localhost/lzop-2-25, lzop level2 x25
processing ./data_25x_localhost/lzop-3-25, lzop level3 x25
processing ./data_25x_localhost/lzop-5-25, lzop level5 x25
processing ./data_25x_localhost/lzop-7-25, lzop level7 x25
processing ./data_25x_localhost/lzop-9-25, lzop level9 x25
```

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
```

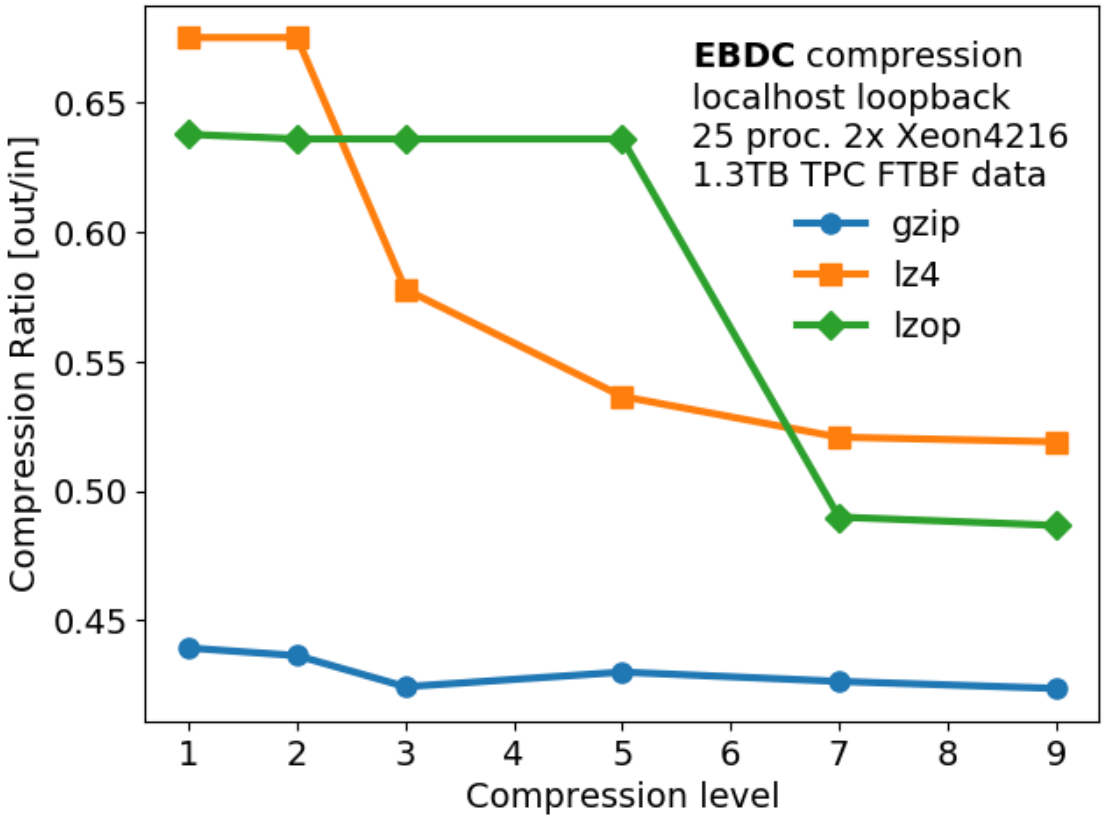
```
processing  lzop . 9  size=  2133 compression ratio =  0.4866962781452209
```

[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

[11]:
```
# 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)
```

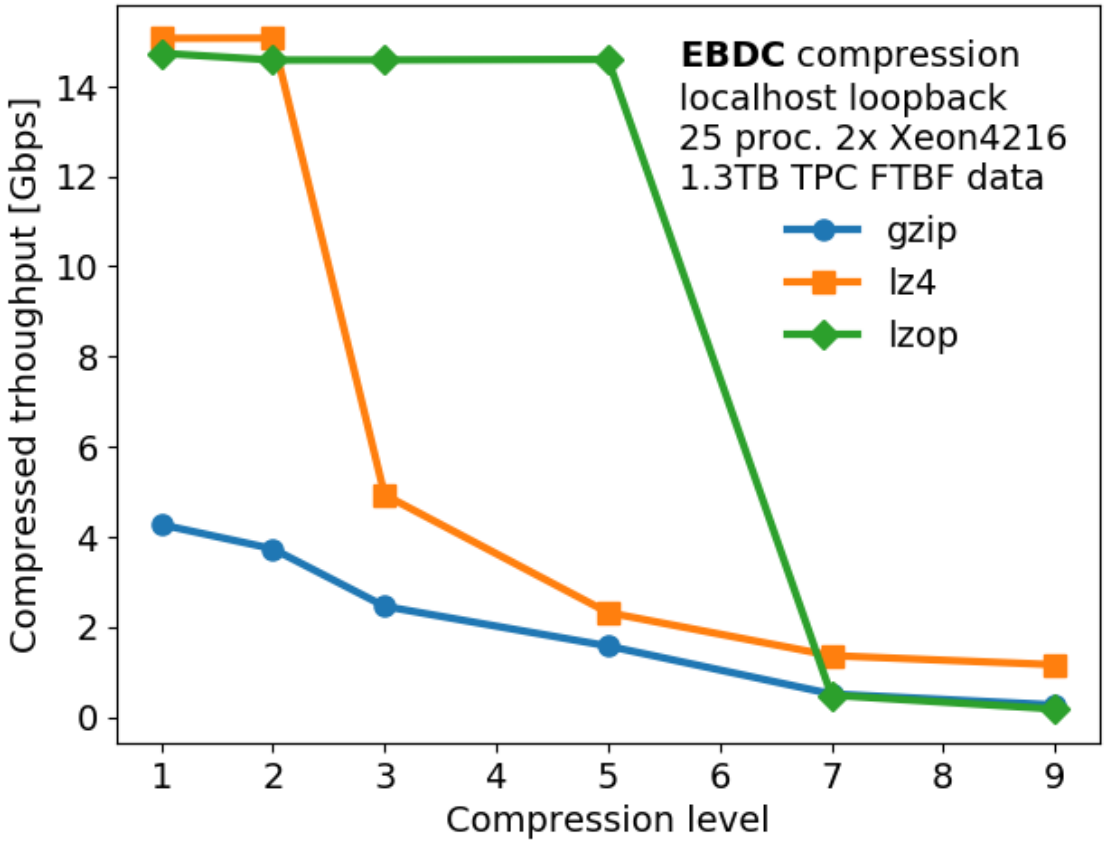
```
plt.savefig(os.path.join(DataDir,"Compression.png"), dpi=150)
plt.savefig(os.path.join(DataDir,"Compression.pdf"), dpi=150)
```



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 trhoughput [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,
```

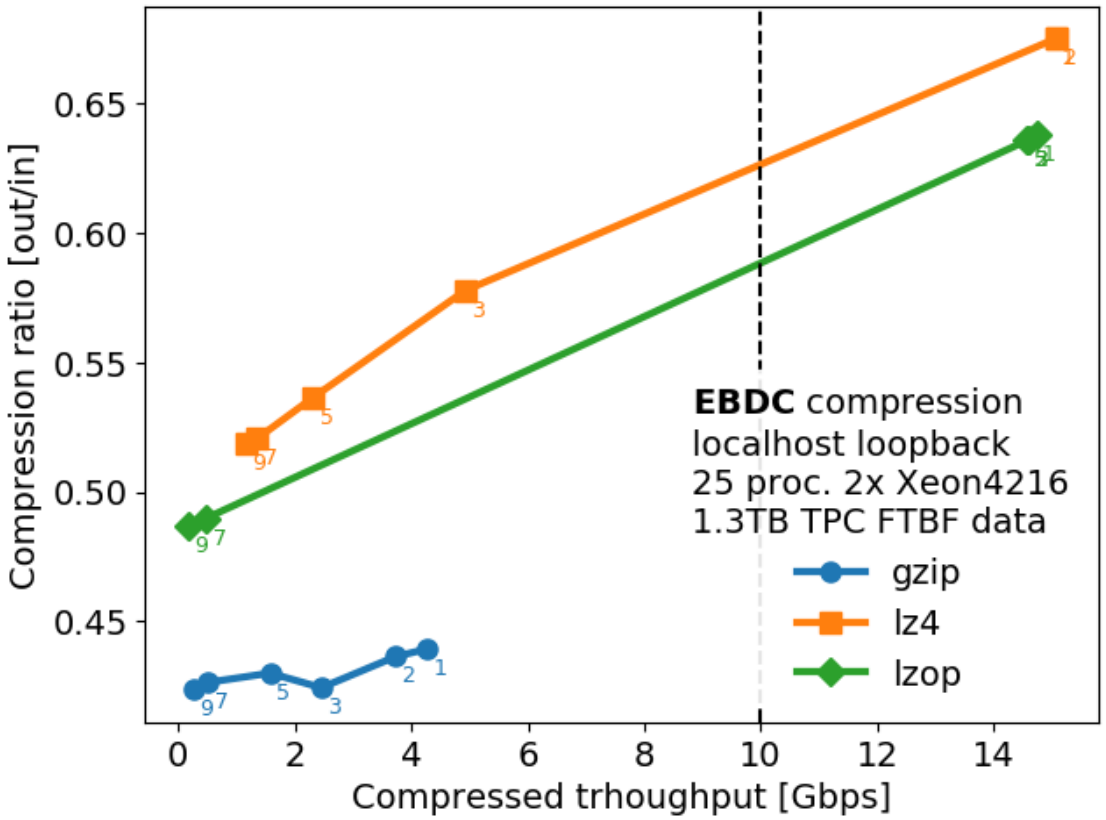
```
            label=zipcmd)

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 trhoughput [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	totalOutSize	Compression	inRateGbps	outRateGbps
0	gzip-1-25	gzip	1	25	27710.4416	1.345599e+12	27711.0109	5.911506e+11	0.439321	9.711858	4.266539
1	gzip-2-25	gzip	2	25	31493.8780	1.345599e+12	31494.5092	5.872109e+11	0.436394	8.545149	3.728973
2	gzip-3-25	gzip	3	25	46565.4287	1.345599e+12	46566.2482	5.710247e+11	0.424365	5.779392	2.452526
3	gzip-5-25	gzip	5	25	73615.0219	1.345599e+12	73616.1849	5.786260e+11	0.430014	3.655774	1.572008
4	gzip-7-25	gzip	7	25	225600.2495	1.345599e+12	225603.1836	5.738001e+11	0.426427	1.192906	0.508681
5	gzip-9-25	gzip	9	25	432304.2058	1.345599e+12	432309.5521	5.701411e+11	0.423708	0.622524	0.263765
6	lz4-1-25	lz4	1	25	12061.4768	1.345599e+12	12064.6152	9.084490e+11	0.675126	22.312349	15.059726
7	lz4-2-25	lz4	2	25	12057.8337	1.345599e+12	12060.9059	9.084490e+11	0.675126	22.319090	15.064358
8	lz4-3-25	lz4	3	25	31553.6753	1.345599e+12	31563.7981	7.775625e+11	0.577856	8.528955	4.926926
9	lz4-5-25	lz4	5	25	62380.4076	1.345599e+12	62400.6044	7.219256e+11	0.536509	4.314173	2.313842
10	lz4-7-25	lz4	7	25	103179.2939	1.345599e+12	103212.2861	7.007271e+11	0.520755	2.608274	1.357837
11	lz4-9-25	lz4	9	25	120706.8721	1.345599e+12	120745.8980	6.983585e+11	0.518994	2.229532	1.156741
12	lzop-1-25	lzop	1	25	11651.7342	1.345599e+12	11652.2921	8.581284e+11	0.637729	23.096980	14.728920
13	lzop-2-25	lzop	2	25	11740.9330	1.345599e+12	11741.5004	8.557885e+11	0.635991	22.921507	14.577158
14	lzop-3-25	lzop	3	25	11739.2630	1.345599e+12	11739.8147	8.557885e+11	0.635991	22.924768	14.579251
15	lzop-5-25	lzop	5	25	11729.0190	1.345599e+12	11729.5595	8.557885e+11	0.635991	22.944790	14.591998
16	lzop-7-25	lzop	7	25	274774.6620	1.345599e+12	274782.5120	6.592038e+11	0.489896	0.979420	0.479800
17	lzop-9-25	lzop	9	25	754100.0086	1.345599e+12	754120.6388	6.548982e+11	0.486696	0.356876	0.173685

[ ]:
```

```