

Bench_214_sq_db70dcc1edb

October 23, 2018

1 Benchmark results from Haswell system

For more information contact @heplesser

```
In [1]: import pandas as pd
import glob
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams['figure.figsize'] = (12, 5)

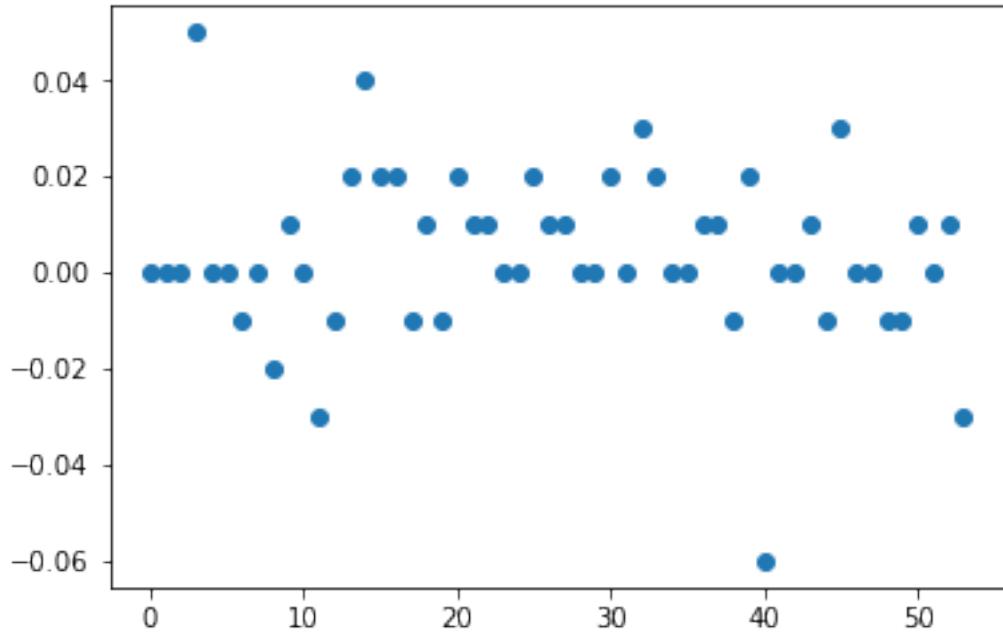
In [2]: codes = {'2.14': 'nest_2.14', 'sq': 'nest_2.16_sq', 'master': 'nest_db70dcc1edb'}
cases = {'s20': 'hpcmam_s20_vp24', 's367': 'hpcmam_s367_mix'}

In [3]: def strip_headers(df):
    df.columns = [col.strip() for col in df.columns]
    return df

In [4]: data = []
for c, d in codes.items():
    for cs in cases.values():
        res = pd.concat((strip_headers(pd.read_csv(f, sep='|', header=1, skiprows=[2])),
                         for f in glob.glob(f'Results/sdv/{d}/{cs}_*.out')),
                         ignore_index=True, sort=False)
        res['Code'] = c
        data.append(res)
res = pd.concat(data, ignore_index=True, sort=False)

In [5]: res['T_bld_xn'] = res['T_conns_min'] + res['T_ini_max']
res['T_bld_nx'] = res['T_conns_max'] + res['T_ini_min']

In [6]: plt.plot(res.T_bld_xn-res.T_bld_nx, 'o');
```



Combining minimum of T_{conn} with maximum of T_{ini} gives consistent build time.

```
In [7]: res['T_bld'] = res[['T_bld_xn', 'T_bld_nx']].min(axis=1)
```

```
In [8]: res[:5]
```

	NUMBER_OF_NODES	TASKS_PER_NODE	THREADS_PER_TASK	SCALE	PLASTIC	\
0	1	24		1	20	
1	1	6		4	20	
2	1	4		6	20	
3	1	3		8	20	
4	1	2		12	20	
	T_nrns	T_conn_min	T_conn_max	T_ini_min	T_ini_max	... N_spks_sum \
0	0.09	31.43	32.43	2.97	3.97	... 2940652
1	0.29	34.12	36.06	3.15	5.09	... 2940652
2	0.44	35.76	36.17	3.18	3.59	... 2940652
3	0.54	34.76	48.34	3.47	17.10	... 2940652
4	0.80	35.65	36.00	3.09	3.44	... 2940652
	Rate_sum	N_nrns	N_connsum	d_min	d_max	Code T_bld_xn T_bld_nx \
0	13.031	225000	1265851000	0.1	50.0	2.14 35.40 35.40
1	13.031	225000	1265851000	0.1	50.0	2.14 39.21 39.21
2	13.031	225000	1265851000	0.1	50.0	2.14 39.35 39.35
3	13.031	225000	1265851000	0.1	50.0	2.14 51.86 51.81
4	13.031	225000	1265851000	0.1	50.0	2.14 39.09 39.09

```

T_bld
0 35.40
1 39.21
2 39.35
3 51.81
4 39.09

[5 rows x 23 columns]

```

```
In [9]: pres = pd.pivot_table(res, aggfunc=np.min, index=['SCALE', 'THREADS_PER_TASK', 'Code'],
                           values=['T_bld', 'T_equ', 'T_sim', 'VSize_sum'])
```

```
In [10]: pres
```

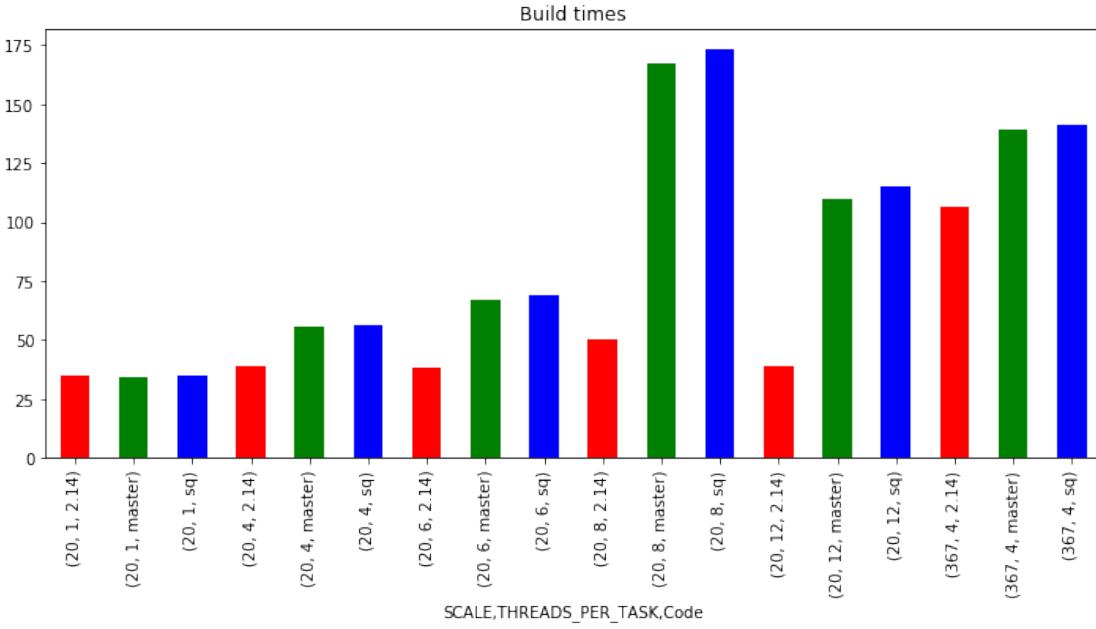
```
Out[10]:
   SCALE THREADS_PER_TASK Code      T_bld    T_equ    T_sim    VSize_sum
20      1                 2.14    35.07    7.57  85.53  44388288
          master        34.25    8.24  93.55 105545952
          sq            34.80    8.28  93.77 44457984
        4                 2.14    38.89    7.55  85.04 42113664
          master        55.54    8.64  97.83 102708936
          sq            56.31    8.71  97.96 42437264
       6                 2.14    38.46    7.75  87.73 41812800
          master        67.03    8.52  96.49 102398832
          sq            68.76    8.54  96.70 43120800
     8                 2.14    50.24    8.29  93.15 41718048
          master       167.46    9.61 108.44 102259524
          sq            173.00    9.69 109.17 42699048
    12                 2.14    38.78    7.88  90.14 42237952
          master       109.68    8.83  99.99 102153240
          sq            115.07    8.88 100.72 42564272
  367      4                 2.14   106.08   24.14 275.48 796302720
          master       138.80   28.37 322.89 1282539456
          sq            141.12   28.88 330.96 831954432
```

1.1 Plots

- Red: NEST 2.14 release version
- Green: Master @ db70dcc1edb (essentially 2.16.0)
- Blue: @hakonsbm's branch off db70dcc1edb with "seque" data structure

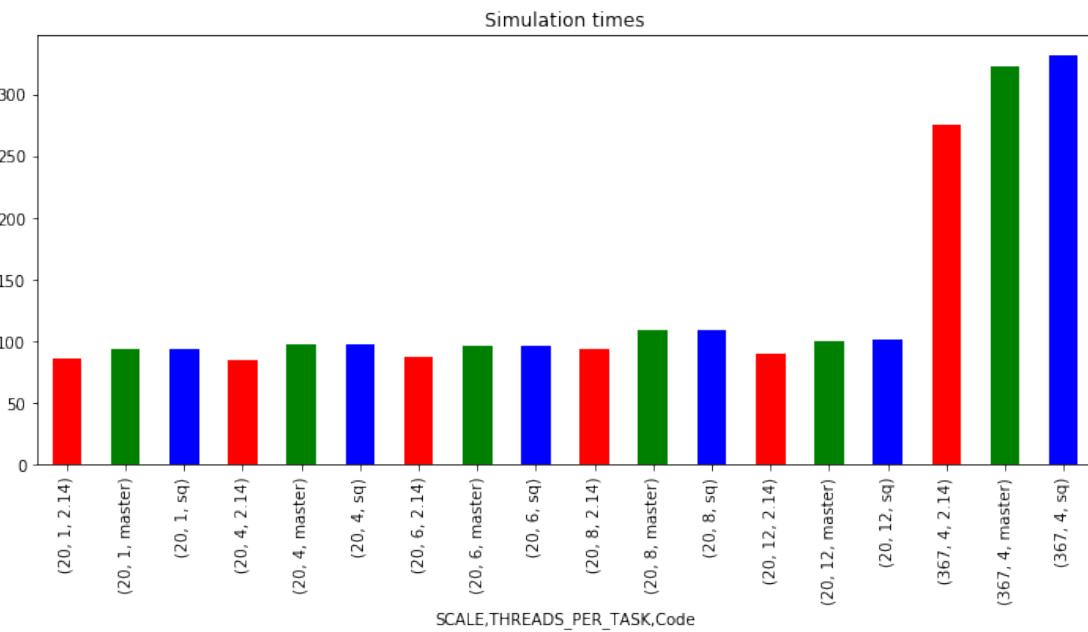
1.1.1 Network construction time

```
In [11]: pres.T_bld.plot(kind='bar', title='Build times', color=['r', 'g', 'b']*int(len(pres)/3),
                        figsize=(12, 5));
```



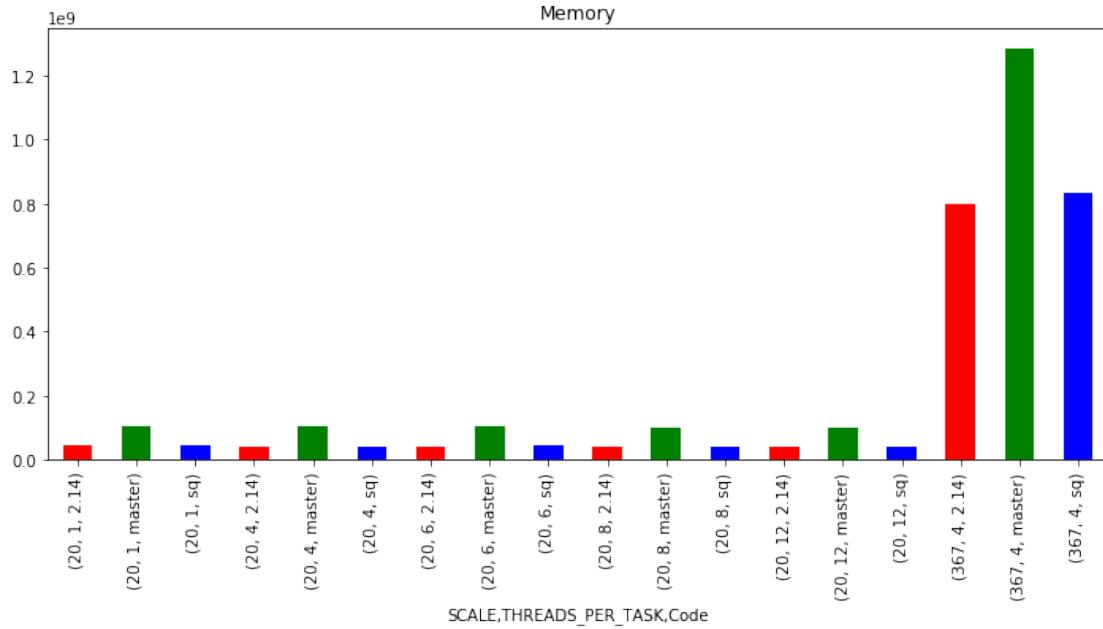
1.1.2 Simulation time

```
In [12]: pres.T_sim.plot(kind='bar', title='Simulation times', color=['r', 'g', 'b']*int(len(pres.T_sim)), figsize=(12, 5));
```



1.1.3 Memory consumption

```
In [13]: pres.VSize_sum.plot(kind='bar', title='Memory', color=['r', 'g', 'b']*int(len(pres)/3), figsize=(12, 5));
```



1.2 Conclusions

- Memory consumption with Seque very close to 2.14
- Significant jump in build times from 2.14 to 2.16
- Some increase from 2.16 when using Seque
- Similar for simulation times
- No difference in build times with 24 MPI process @ 1 thread

1.2.1 Real MAM on different Haswell-based system

- Total runtime including 2000ms simulation
 - 4g master in June 2018, 720 MPI proc @ 1 thread: ca 1 hour
 - db70dcc1edb runs over 1 hour (killed) with 720 x 1, but just about **20 minutes** with 180 x 4
 - very similar results with Seque

1.3 Recommendations

- Seque is ready for use
- We need to understand the thread-dependency of run times