

ctf

October 28, 2019

```
In [2]: import json
import pandas as pd
import seaborn as sns; sns.set()
import babeltrace as bl
```

```
In [3]: tr = bl.TraceCollection()
tr.add_trace('sample_traces/jsoo/ctf/', "ctf")
```

```
events = []
counters = []
allocs = []
```

```
ENTRY = 0
EXIT = 1
COUNTER = 2
ALLOC = 3
```

*# babeltrace is an evil binding, do not put Event objects directly in a list, need to use
Events objects are actually references to the head of the iterator, so trying to use
the scope of the iterator will results in having the same objects, that is, the last*

```
for event in tr.events:
    id = event['id']
    ev = {}
    ev['timestamp'] = event['timestamp']
    if id == ENTRY or id == EXIT:
        ev['phase'] = event['phase']
        ev['id'] = id
        events.append(ev)
    if id == COUNTER:
        ev['name'] = event['kind']
        ev['count'] = event['count']
        counters.append(ev)
    if id == ALLOC:
        ev['bucket'] = event['bucket']
        ev['count'] = event['count']
        allocs.append(ev)
```

```

timeline_set = {}
timeline = []

for event in events:
    phase = event['phase']
    if event['id'] == ENTRY:
        if phase in timeline_set:
            print("overlapping entry for phase: %s, timestamp %d" % (phase, event['timestamp']))
        else:
            timeline_set[phase] = event
    elif event['id'] == EXIT:
        if phase in timeline_set:
            ev_entry = timeline_set[phase]
            item = {
                'ts_begin': ev_entry['timestamp'],
                'ts_end': event['timestamp'],
                'dur': event['timestamp'] - ev_entry['timestamp'],
                'phase': phase
            }
            timeline.append(item)
            del timeline_set[phase]
        else:
            print("got an exit event with no entry: %s, timestamp %d" % (phase, event['timestamp']))

for key in timeline_set:
    print("got an entry event with no exit: %s, timestamp: %d" % (key, timeline_set[key]['timestamp']))

```

In [4]: *# From here, refer to <https://github.com/Engil/gc-tracing/blob/master/metadata> to know the various phases and counters available.*

```

def counters_by_name(name): return [ev for ev in counters if ev['name'] == name]
def events_by_name(name): return [ev for ev in timeline if ev['phase'] == name]

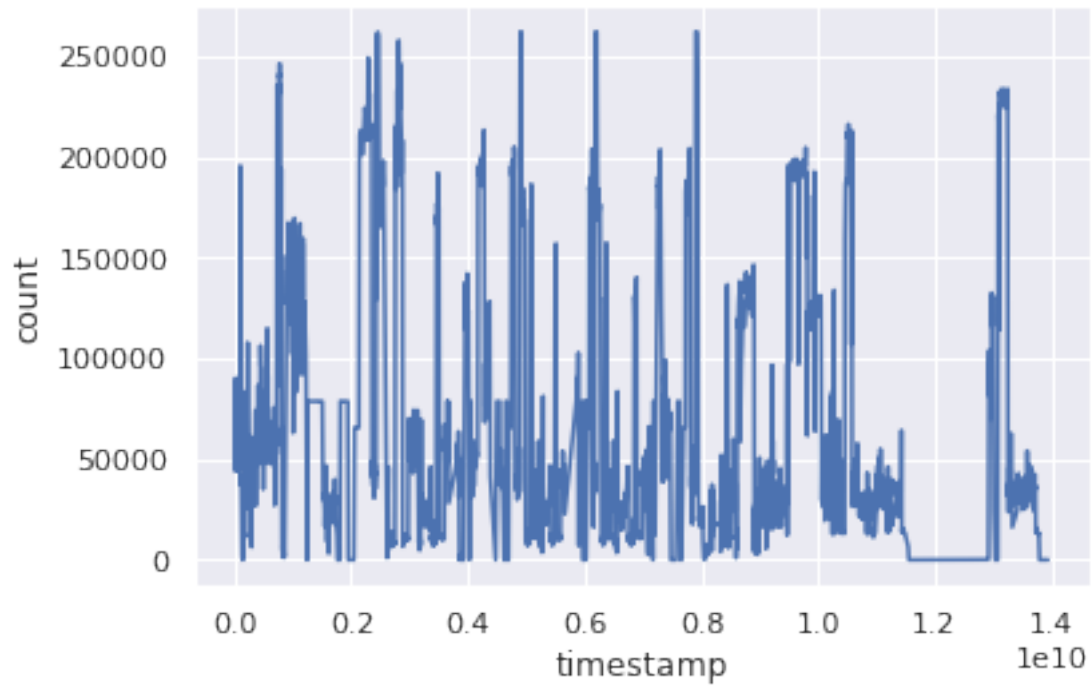
s = pd.DataFrame(counters_by_name('minor/promoted'))

s['timestamp'] = s['timestamp'].astype('timedelta64[ns]')

sns.lineplot(x="timestamp", y="count", data=s)

```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f88aea320>

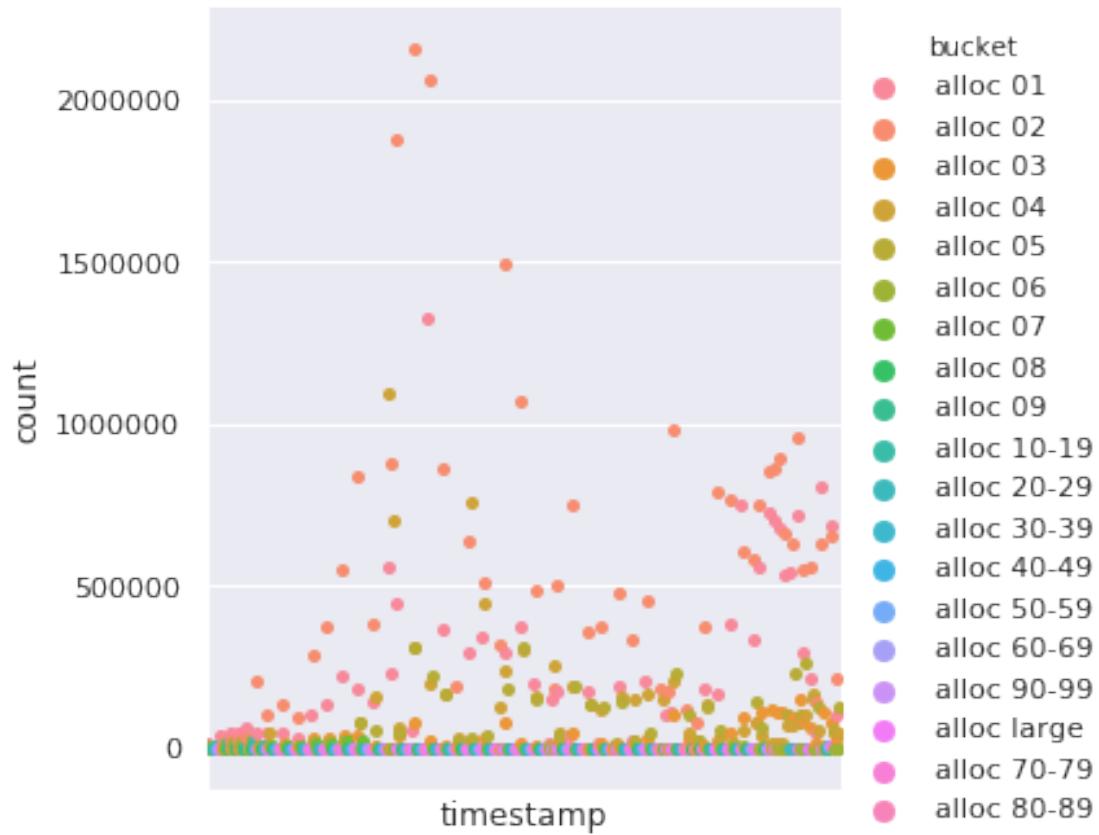


In [5]: # Visiting allocations during the lifetime of the program, by buckets

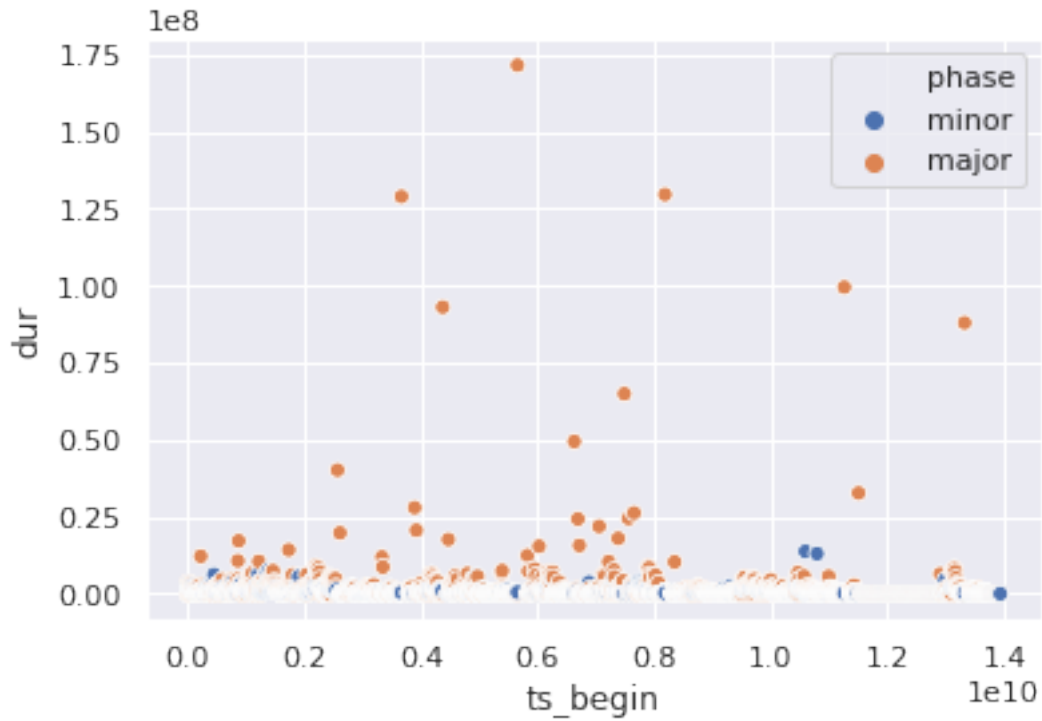
```
s = pd.DataFrame(allocs)
```

```
ax = sns.catplot(x='timestamp', y='count', hue='bucket', data=s)  
ax.set(xticks=[])
```

Out [5]: <seaborn.axisgrid.FacetGrid at 0x7f4fd562a160>



```
In [7]: # minor and major dispatches, duration on the Y axis
s = pd.DataFrame([ev for ev in timeline if ev['phase'] == 'major' or ev['phase'] == 'm
ax = sns.scatterplot(x='ts_begin', y='dur', hue='phase', data=s)
```



```
In [8]: def gc_phase_quantile(name):
         s = pd.DataFrame(events_by_name(name))
         return (s['dur'].quantile([.1, .25, .50, .75, .90, .95, .99]))
```

```
In [9]: gc_phase_quantile('major')
```

```
Out[9]: 0.10      7624.90
        0.25     95312.50
        0.50    313291.50
        0.75    848809.75
        0.90   1763479.60
        0.95   2827364.80
        0.99   7396957.73
        Name: dur, dtype: float64
```

```
In [10]: gc_phase_quantile('minor')
```

```
Out[10]: 0.10      45899.70
        0.25     96412.75
        0.50    223426.00
        0.75    448376.00
        0.90   1180273.50
        0.95   1578354.75
        0.99   2438961.69
        Name: dur, dtype: float64
```

```
In [11]: gc_phase_quantile('major/sweep')
```

```
Out[11]: 0.10      725.00  
         0.25     69190.25  
         0.50    214389.00  
         0.75    391910.00  
         0.90   1091568.00  
         0.95   1665485.50  
         0.99   3856271.60  
         Name: dur, dtype: float64
```