TooManyShards

May 3, 2018

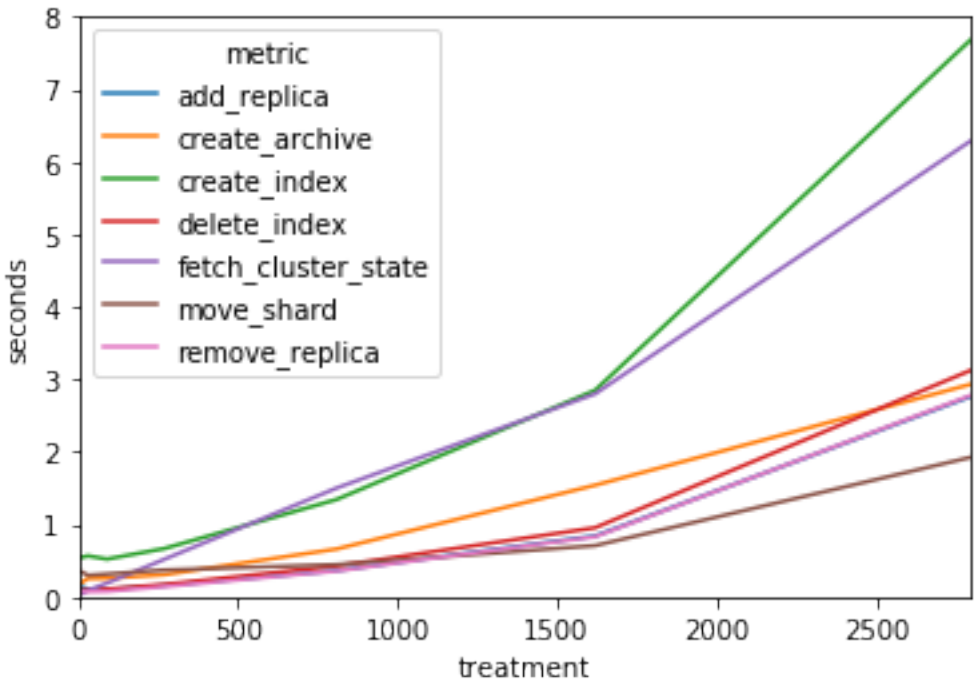
In [62]:
```
import math
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

def read_metrics(ident, path):
    with open(path, 'r') as f:
        for line in f:
            pieces = line.strip().split(':')
            if len(pieces) == 2:
                metric, value = pieces
            elif len(pieces) == 3:
                timestamp, metric, value = pieces
            else:
                raise Exception("Couldn't parse line: %s" % (line))
            yield ident, metric, float(value)

def read_all_metrics(paths):
    def inner():
        for ident, path in paths:
            yield from read_metrics(ident, path)
    df = pd.DataFrame(inner())
    df.columns = ('treatment', 'metric', 'value')
    return df

def reject_outliers(df_in, k=3):
    # k is the number of allowed standard deviations away from the mean
    aggs = df_in.groupby(['treatment', 'metric']).aggregate([np.mean, np.std])
    max_val = aggs['value']['mean'] + (k * aggs['value']['std'])
    max_val.name = 'max_val'
    min_val = aggs['value']['mean'] - (k * aggs['value']['std'])
    min_val.name = 'min_val'
    df_out = df_in.join(min_val, on=['treatment', 'metric']).join(max_val, on=['treatm
    condition = (df_out['min_val'] < df_out['value']) & (df_out['value'] < df_out['max
    return df_out[condition][['treatment', 'metric', 'value']]

def report(path_format, n_indices):
```

```
    df = read_all_metrics((index_count, path_format.format(index_count)) for index_cou
    df_agg = df.groupby(['metric', 'treatment']).mean().reset_index() \
           .pivot(index='treatment', columns='metric', values='value')
    # Data collection was bad and included fetch_cluster_state in move_shard collecti
    df_agg['move_shard'] -= df_agg['fetch_cluster_state']
    # Not meaningful like this
    del df_agg['dupe']
    ax = df_agg.plot(xlim=(0,2800), ylim=(0, 8))
    ax.set_ylabel('seconds')

def compare(*paths, max_std=None):
    df = read_all_metrics((x, x) for x in paths)
    if max_std is not None:
        df = reject_outliers(df, max_std)
    df_agg = (df[df['metric'] != 'dupe']
        .groupby(['treatment', 'metric'])
        .aggregate([np.min, np.mean, np.max, np.std])
        .unstack('metric').transpose()
        .reset_index(level=0, drop=True)
        .rename_axis(['aggregation', 'metric'])
        .reset_index()
        .sort_values(['metric', 'aggregation'])
        .set_index(['metric', 'aggregation']))
    return df_agg

def detail_report(path):
    df = read_all_metrics([(path, path)])
    metrics = list(df['metric'].unique())
    if 'create_archive' in metrics:
        metrics.remove('create_archive')
        metrics.append('create_archive')
    height = int(math.ceil(len(metrics)/2.0))
    fig, axes = plt.subplots(height, 4, figsize=(16, 3 * height))
    fig.suptitle(path)
    for i, metric in enumerate(metrics):
        x = i // 2
        y = 0 if i % 2 == 0  else 2
        df_metric = df[df['metric'] == metric].reset_index(drop=True)
        axes[x][y].set_title('{} hist'.format(metric))
        df_metric.value.plot.hist(ax=axes[x][y])
        axes[x][y + 1].set_title('{} over time'.format(metric))
        axes[x][y + 1].set_ylabel('seconds')
        df_metric.value.plot(ax=axes[x][y + 1])
    plt.tight_layout()
    plt.subplots_adjust(top=0.90)
```

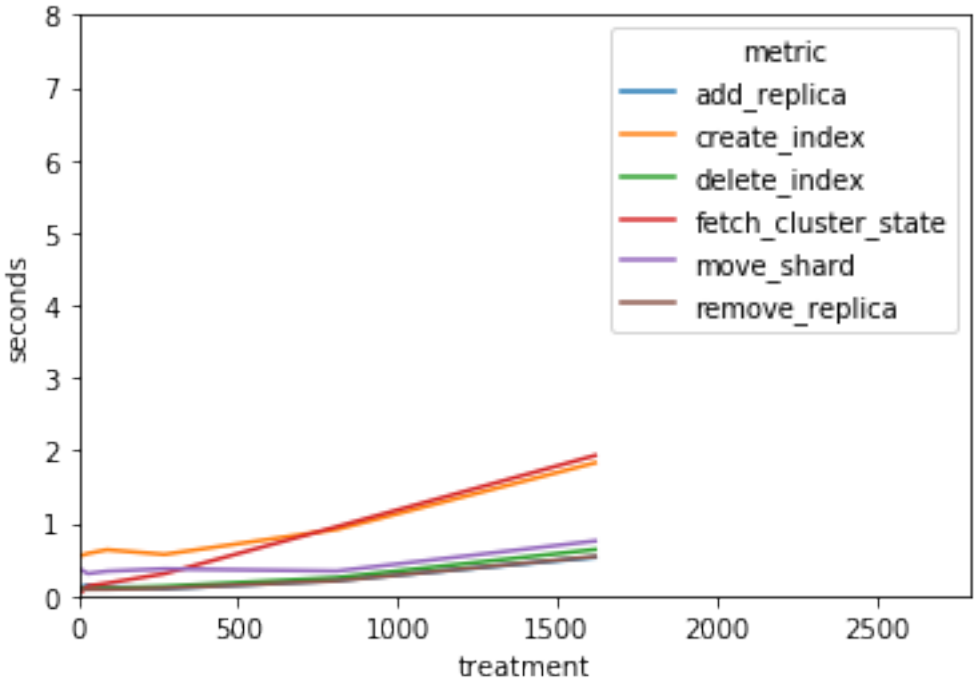
0.1 4 node cluster on laptop

These first two graphs are from a 4 node cluster running on a laptop to check initial feasability and develop measurement software. They y axis is the mean number of seconds to complete an operation. The x axis is the number of non-archive indices that exist. The first graph has, in addition to the specified number of indices, an expected (1/3 of index count) number of additional archive indices that we are measuring the impact of adding.

```
In [63]: report('latencies-archive-{}', (10, 30, 90, 270, 810, 1620, 2798))
```




```
In [64]: report('latencies-{}', (10, 30, 90, 270, 810, 1620))
```



0.2 Compare production clusters

Compare latencies of various operations on the clusters under different treatments. codfw tests run from wasat. eqiad tests run from terbium. Minor changes in the data collection were made between runs. all values reported here are in seconds.

Race conditions (and sub-par data collection) mean that sometimes actions that cause the cluster to go green->yellow, such as adding a replica, are recorded into the next action that waits for green.

eqiad-with-archive-again is a re-run of eqiad-with-archive after moving busy shards away from the master node. The high level idea was to see if having a more idle (or perhaps in the future dedicated) master makes any difference on these operations. It still suffers some of the same problems going green after creating a new index, likely those are un related to master load. Additionally this treatment does not report create_archive because the archive indices created for eqiad-with-archive had not been deleted yet.

```
In [65]: compare('codfw-with-archive', 'codfw-default', 'eqiad-default', 'eqiad-with-archive',
```

```
Out[65]: treatment                        codfw-default  codfw-with-archive  \
         metric              aggregation
         add_replica         amax               1.802325            1.801499
                             amin               0.421666            0.541755
                             mean               0.594346            0.741316
                             std                0.227375            0.227987
         create_archive      amax                    NaN            9.565032
                             amin                    NaN            0.703246
```

	mean	NaN	1.490510
	std	NaN	0.547086
create_index	amax	9.815382	9.906802
	amin	1.083011	1.526315
	mean	1.785033	2.417510
	std	0.980366	0.991928
delete_index	amax	2.472375	3.236383
	amin	0.389096	0.491986
	mean	0.649102	0.953553
	std	0.295380	0.531001
fetch_cluster_state	amax	9.661135	8.112589
	amin	5.059470	6.433290
	mean	5.796606	7.238008
	std	0.568421	0.393194
move_shard	amax	2.279626	2.914373
	amin	0.504794	0.713125
	mean	1.511768	1.995401
	std	0.214433	0.288843
remove_replica	amax	1.135170	1.577174
	amin	0.387481	0.483768
	mean	0.531659	0.711126
	std	0.152439	0.202469

treatment		eqiad-default	eqiad-with-2x-archive	\
metric	aggregation			
add_replica	amax	4.585436	10.201376	
	amin	1.117231	1.171589	
	mean	1.684832	3.138669	
	std	0.767619	1.873895	
create_archive	amax	NaN	13.736601	
	amin	NaN	1.479052	
	mean	NaN	3.300480	
	std	NaN	1.049914	
create_index	amax	7.342426	673.694876	
	amin	2.125578	3.325205	
	mean	3.402258	21.817991	
	std	1.069743	72.191530	
delete_index	amax	3.821631	10.051981	
	amin	0.922939	1.083179	
	mean	1.602840	3.509121	
	std	0.636497	2.257078	
fetch_cluster_state	amax	12.322482	14.915290	
	amin	6.124049	9.096656	
	mean	7.010527	10.859584	
	std	0.726586	0.945218	
move_shard	amax	5.109589	11.562633	
	amin	1.584221	1.370179	
	mean	2.641831	4.802176	

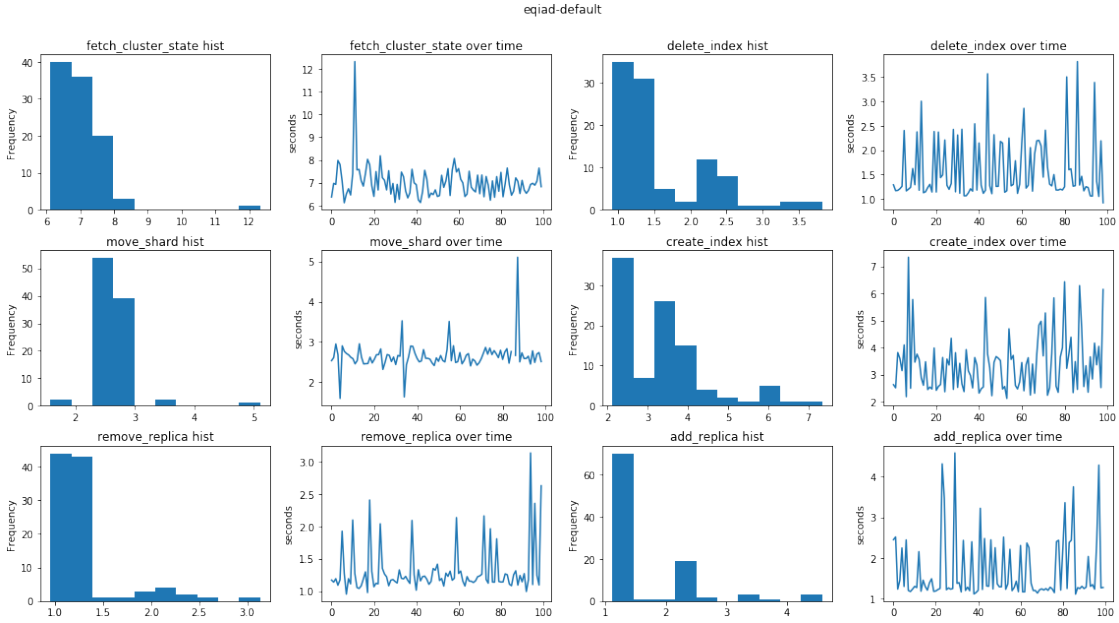
	std	0.346483	1.805751
remove_replica	amax	3.135641	6.402294
	amin	0.955662	1.016324
	mean	1.308679	2.576980
	std	0.372522	1.377775

treatment		eqiad-with-archive	eqiad-with-archive-again
metric	aggregation		
add_replica	amax	97.446347	219.807150
	amin	1.184394	0.952167
	mean	4.231860	4.580660
	std	9.590064	21.944165
create_archive	amax	13.576228	NaN
	amin	1.150909	NaN
	mean	3.074341	NaN
	std	0.931940	NaN
create_index	amax	323.704661	90.748019
	amin	2.887910	0.093981
	mean	13.629758	6.734137
	std	36.401288	9.191905
delete_index	amax	12.737284	10.542377
	amin	1.042131	0.909876
	mean	3.876121	2.739332
	std	2.502692	1.903640
fetch_cluster_state	amax	12.267843	14.648425
	amin	7.674957	7.416046
	mean	8.986889	9.251392
	std	0.784973	1.036539
move_shard	amax	9.215395	8.243102
	amin	2.552566	1.115686
	mean	4.430160	3.278687
	std	1.133049	1.013169
remove_replica	amax	11.275946	11.271543
	amin	0.987701	0.791463
	mean	2.878008	1.697925
	std	1.676537	1.258968

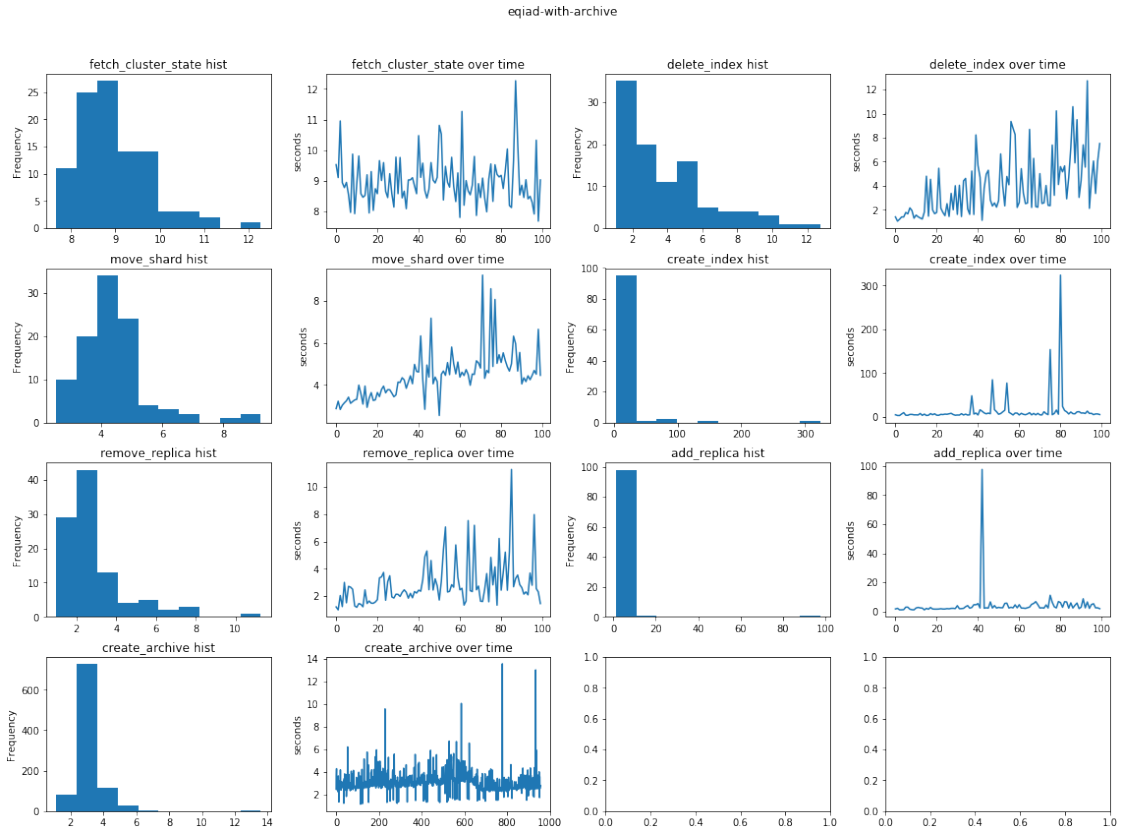
0.3 Detailed per-run reports

The rest of the graphs here show details about test runs in terms of a histogram and an over-time graph per metric

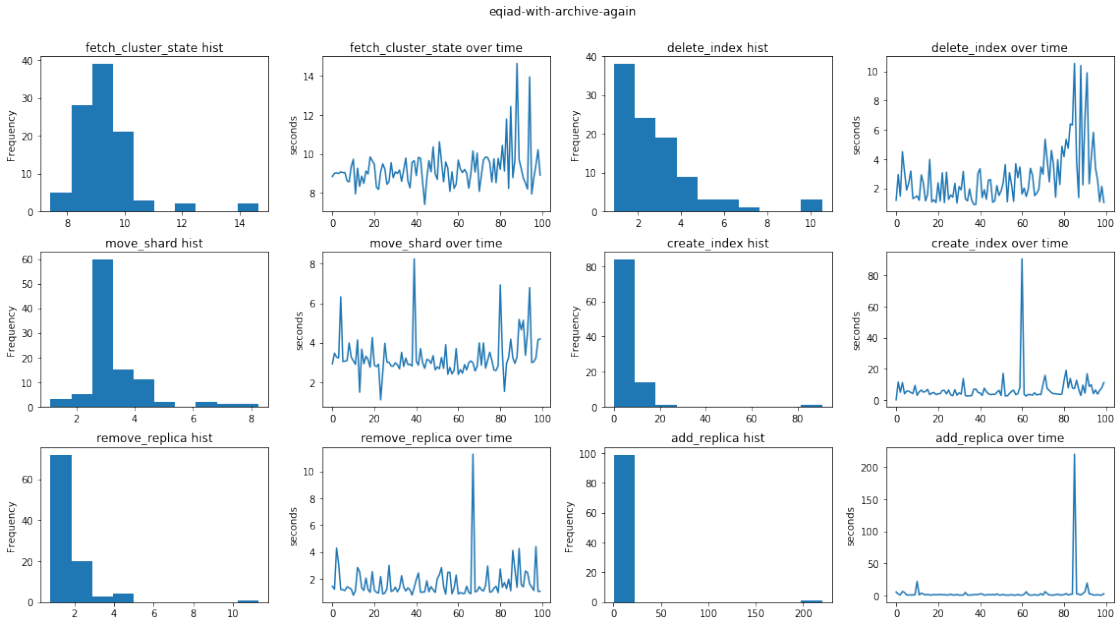
```
In [66]: detail_report('eqiad-default')
```



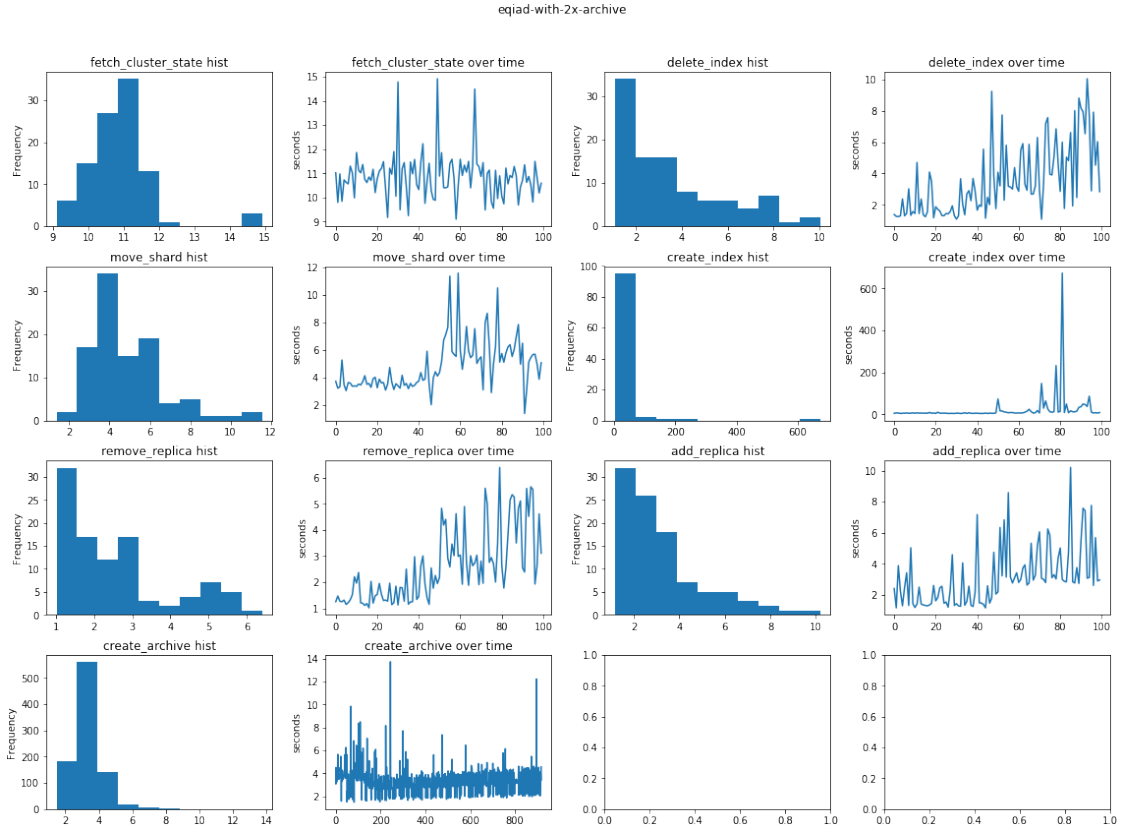
In [67]: `detail_report('eqiad-with-archive')`



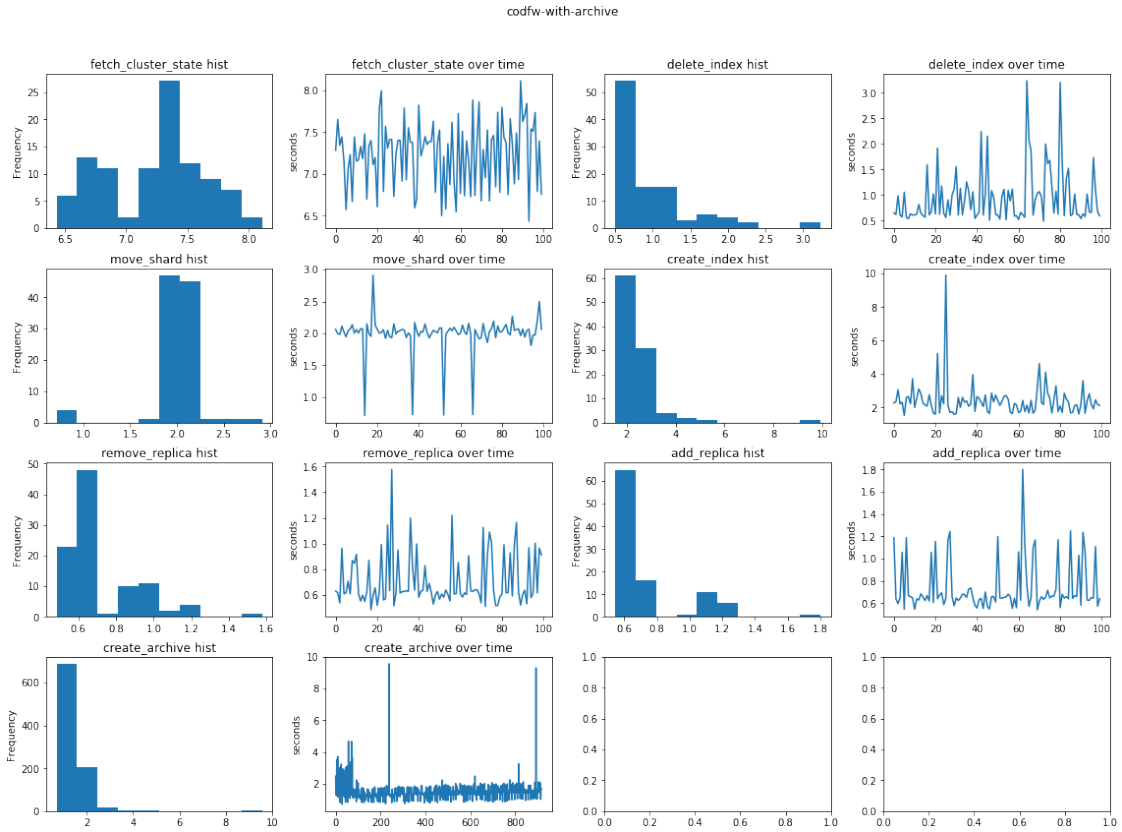
In [68]:
```
# This particular report was run by moving busy shards away from the master node. Thi
# the master idle but it did cut load in half.
detail_report('eqiad-with-archive-again')
```



In [69]:
```
detail_report('eqiad-with-2x-archive')
```



In [70]: `detail_report('codfw-with-archive')`



0.4 Methods

Data was collected via a python script. Six cluster mutation actions were chosen and repeated 100 times. For each of the 100 iterations the order of actions was randomized. All operations that mutate an index were limited to empty indices with no documents created for the purpose of this test.

0.4.1 Setup

create_archive Prior to running the test we create archive indices to match all of the general indices that currently exist. This is only reported on tests that created archives, baseline tests do not report this number (although they do create 10 archive indices so the mutation tests have something to work with)

0.4.2 Cluster State Mutation Tests

fetch_cluster_state Measure seconds taken for cluster to respond to /_cluster/state

delete_index Measure seconds taken for cluster to respond to DELETE request on /{index}

move_shard Measure seconds taken for cluster to respond to POST on /_cluster/reroute moving shard between nodes followed by a wait for /{index}/_recovery to report that shard has finished moving

create_index Measure seconds taken for cluster to respond to PUT to /{index} and for the cluster to return to a green state. mappings and settings are copied from a random index that already exists on the cluster.

remove_replica Measure sceonds taken for cluster to respond to PUT on /{index}/_settings with reduced auto_expand_replicas

add_replica Measure seconds taken for cluster to respond to PUT on /{index}/_settings with increased auto_expand_replicas followed by a wait for cluster to return to green