ApplianceTelemetryCorrelationAnalysis

October 20, 2024

0.1 Provided as-is (w/o support)

Kubernetes clusters collect various application and infrastrcuture statistics. While this information is useful, it's very difficult to identify which metrics are useful for monitoring and troubleshooting. The Goal here is to collect this information, and use a statistical model to identify which metrics should be included in reports/dashboard such that: * Unnecessary overhead and sensory overload can be reduced. * Time can be saved by prioritising monitoring the correct metrics.

This process needs to assume `zero knowlidge` of the workings of the cluster, workload being run and any other information. This way, `generic` clusters can be monitored without explicitly programming dashboards based on internal knowlidge. This is also a good method to discover/verify application knowlidge/bottlenecks with statistical data analyis.

0.2 Step1: Data Loading

We will load cpu, memory, task_queue information along with stats from strcutured and unstrucutred scans from csv files stored on disk using the `dataframeLoader` helper.

```
# The dataframeLoader helper function implements the loadApplianceTimeSeriesData method.
# This method loads the csv files, and pivots them to generate distinct "metrics" timeseries.
# see https://github.com/amitgupta7/docker-jupy-ntbk-s3-reporting/blob/main/dataframeLoader.py
```

[1]:
```
import sys
sys.path.append('../')

import dataframeLoader as dfl
import pandas as pd
from importlib import reload
reload(dfl)

# Provide csv data location and appliance and timerange information.
root = '../../.dataDir'
fromDt = '2024-06-01'
toDt = '2024-11-01'

# Provide list of prometheus metrics to load.
metricsArr = ['cpu_used', 'download_workers_count', 'memory_used',
 ↪'task_queue_length', 'infra_access_latency', 'pod_cpu_usage',
 ↪'pod_memory_usage']
# metricsArr = ['cpu_used'
```

```
#               ,'task_queue_length'
#               , 'memory_used'
#               ]

daterange=[fromDt, toDt]
df = dfl.loadApplianceTimeSeriesData(root, metricsArr, daterange)
```

```
loading Unstrctured Data from file: SCANPROC-*.csv

loading Strctured Data from file: STRUCTURED-*.csv

processing securiti_appliance_cpu_used-max*.csv

processing securiti_appliance_cpu_used-avg*.csv

processing securiti_appliance_download_workers_count-max*.csv

processing securiti_appliance_download_workers_count-avg*.csv

processing securiti_appliance_memory_used-max*.csv

processing securiti_appliance_memory_used-avg*.csv

processing securiti_appliance_task_queue_length-max*.csv

processing securiti_appliance_task_queue_length-avg*.csv

processing securiti_appliance_infra_access_latency-max*.csv

processing securiti_appliance_infra_access_latency-avg*.csv

processing securiti_appliance_pod_cpu_usage-max*.csv

processing securiti_appliance_pod_cpu_usage-avg*.csv

processing securiti_appliance_pod_memory_usage-max*.csv

processing securiti_appliance_pod_memory_usage-avg*.csv

loading Unstrctured Data from file: UNSTRUCTURED-*.csv
```

0.3 Step2: Data Pivoting

We now aggregate the data by `appliance_id` (unique identifier for our cluster) and `ts` timestamp, to get different metrics values as separate columns. Notice there are: * Statistical significance (consider correlation for appliances with atleast 30 non-zero scan time values, ie the appliance should be scanning for atleast 30 hours before considering it statistically significant). * 200+ appliances * Total scanning time of over 15 years!!

- 33 metrics -> 22 metrics
  - Decide between `max` or `avg` values if both are present.
  - We chose to display `avg` values metrics in this case after some trial and error.
    - Except for memory (where max indicates spikes/oom conditions better)
- Tracked every hour

[2]:
```python
#consider only appliances with a certain number of scanning intervals.
min_scanning_intervals = 10
dfp = df.pivot_table(index=['appliance_id','ts'], columns=['metrics'],␣
 ↪values='value', aggfunc='sum').reset_index()
max_list = list(dfp.filter(regex='max'))
# maxlist = ['cpu_used_max', 'linkerq_max', 'memory_used_avg', 'taskq_max',␣
 ↪'tmp_taskq_max']
# we would like to use max memory (to indicate spikes/oom conditions)
max_list = list(map(lambda x: x.replace('memory_used_max', 'memory_used_avg'),␣
 ↪max_list))
print('cols removed', max_list)
dfp = dfp[dfp.columns.drop(max_list)]
stat_sig = dfp.fillna(0).groupby(['appliance_id']).scanTime.agg(lambda x: x.
 ↪ne(0).sum())
stat_sig = stat_sig[stat_sig > min_scanning_intervals].
 ↪sort_values(ascending=False)
print('cols retained', dfp.columns)
print(len(stat_sig), 'statistically significant appliances with total scan time␣
 ↪of', stat_sig.sum()/24/365, 'years')
dfp = dfp[dfp.appliance_id.isin(stat_sig.index)]

display(dfp)
```

```
cols removed ['cpu_used_max', 'download_workers_count_max', 'esLatency_max',
'linkerq_max', 'memory_used_avg', 'pgLatency_max', 'pod_cpu_usage_max',
'pod_memory_usage_max', 'redisLatency_max', 'taskq_max', 'tmp_taskq_max']
cols retained Index(['appliance_id', 'ts', 'IdleTimeInHrs', 'avgFileSizeInMB',
       'cpu_used_avg', 'dataScannedinGB', 'download_workers_count_avg',
       'esLatency_avg', 'fileDownloadTimeInHrs', 'linkerq_avg',
       'memory_used_max', 'numFilesScanned', 'numberOfChunksScanned',
       'numberOfColsScanned', 'pgLatency_avg', 'pod_cpu_usage_avg',
       'pod_memory_usage_avg', 'redisLatency_avg', 'scanTime', 'taskq_avg',
       'tmp_taskq_avg', 'uniqPodCount'],
      dtype='object', name='metrics')
277 statistically significant appliances with total scan time of
15.499086757990868 years
```

metrics	appliance_id	ts	\
57	01c75278-9c0d-41be-b693-c970b18dbedc	2024-06-01 00:00:00	
58	01c75278-9c0d-41be-b693-c970b18dbedc	2024-06-01 01:00:00	
59	01c75278-9c0d-41be-b693-c970b18dbedc	2024-06-01 02:00:00	
60	01c75278-9c0d-41be-b693-c970b18dbedc	2024-06-01 03:00:00	
61	01c75278-9c0d-41be-b693-c970b18dbedc	2024-06-01 04:00:00	
…	…	…	
848488	ffc2fb2f-52e0-42f3-8564-9545dbc6b747	2024-08-21 00:00:00	
848489	ffc2fb2f-52e0-42f3-8564-9545dbc6b747	2024-08-21 01:00:00	
848490	ffc2fb2f-52e0-42f3-8564-9545dbc6b747	2024-08-21 02:00:00	
848491	ffc2fb2f-52e0-42f3-8564-9545dbc6b747	2024-08-21 03:00:00	

```
848492   ffc2fb2f-52e0-42f3-8564-9545dbc6b747 2024-08-21 04:00:00

metrics  IdleTimeInHrs  avgFileSizeInMB  cpu_used_avg  dataScannedinGB  \
57            2.396345         0.036537           NaN         0.001096
58            1.804338         0.851661           NaN         0.034066
59            5.472733         0.067678           NaN         0.006091
60            2.701624         6.814020           NaN         0.293003
61            3.396194         0.304974           NaN         0.015249
…                  …              …             …               …
848488             NaN              NaN      2.723333              NaN
848489             NaN              NaN      2.723333              NaN
848490             NaN              NaN      2.723333              NaN
848491             NaN              NaN      2.723333              NaN
848492             NaN              NaN      2.723333              NaN

metrics  download_workers_count_avg  esLatency_avg  fileDownloadTimeInHrs  \
57                              NaN            NaN                    NaN
58                              NaN            NaN                    NaN
59                              NaN            NaN                    NaN
60                              NaN            NaN                    NaN
61                              NaN            NaN                    NaN
…                             …            …                    …
848488                          NaN       0.000603                    NaN
848489                          NaN       0.000603                    NaN
848490                          NaN       0.000603                    NaN
848491                          NaN       0.000603                    NaN
848492                          NaN       0.000603                    NaN

metrics  linkerq_avg  …  numberOfChunksScanned  numberOfColsScanned  \
57               NaN  …                    NaN                  NaN
58               NaN  …                    NaN                  NaN
59               NaN  …                    NaN                  NaN
60               NaN  …                    NaN                  NaN
61               NaN  …                    NaN                  NaN
…              …  …                  …                …
848488           NaN  …                    NaN                  NaN
848489           NaN  …                    NaN                  NaN
848490           NaN  …                    NaN                  NaN
848491           NaN  …                    NaN                  NaN
848492           NaN  …                    NaN                  NaN

metrics  pgLatency_avg  pod_cpu_usage_avg  pod_memory_usage_avg  \
57                 NaN                NaN                   NaN
58                 NaN                NaN                   NaN
59                 NaN                NaN                   NaN
60                 NaN                NaN                   NaN
61                 NaN                NaN                   NaN
…                  …               …                   …
```

```
848488        0.056978         4.813333            36.355
848489        0.056978         4.813333            36.355
848490        0.056978         4.813333            36.355
848491        0.056978         4.813333            36.355
848492        0.056978         4.813333            36.355

metrics  redisLatency_avg  scanTime  taskq_avg  tmp_taskq_avg  uniqPodCount
57                    NaN  0.029513        NaN            NaN           3.0
58                    NaN  0.195418        NaN            NaN           3.0
59                    NaN  0.057962        NaN            NaN           4.0
60                    NaN  0.196190        NaN            NaN           3.0
61                    NaN  0.035388        NaN            NaN           3.0
…                       …       …       …             …          …
848488           0.003052       NaN        NaN            NaN           NaN
848489           0.003052       NaN        NaN            NaN           NaN
848490           0.003052       NaN        NaN            NaN           NaN
848491           0.003052       NaN        NaN            NaN           NaN
848492           0.003052       NaN        NaN            NaN           NaN

[470148 rows x 22 columns]
```

0.4 Step 3: Data transformation and correlation

We need to acheve two main goals: 1. Isolate data for individual appliance. 2. Remove ghost correlation between unrelated metrics. * We will calculate percentage change between adjacent timeseries values. 3. Calculate absolute correlation between metrics for each single appliance. * Transpose every `metrics` corelation. 4. Generate correlation for every `appliance_id` and `metric` identifier using steps 1, 2 and 3

[3]:
```python
# appliance = '01c75278-9c0d-41be-b693-c970b18dbedc'
# for metric in metrics_category_order:
dfc_arr = []
for pod in dfp.appliance_id.unique():
    dfa = dfp[(dfp.appliance_id == pod)]
    dfa = dfa.drop(['appliance_id', 'ts'], axis=1)
    dfa = dfa.pct_change(periods=1, fill_method=None)
    dfca = dfa.corr().abs()
    # print(type(dfca))
    for col in dfca.columns:
        # print(col)
        dfc = dfca[col].to_frame().T
        dfc.insert(0, 'metric', col )
        dfc.insert(0, 'appliance_id', pod )
        dfc_arr.append(dfc)
dfc = pd.concat(dfc_arr, ignore_index=True)
dfc.set_index('appliance_id', inplace=True)
dfc.head()
```

```
[3]: metrics                                                   metric  \
appliance_id
01c75278-9c0d-41be-b693-c970b18dbedc               IdleTimeInHrs
01c75278-9c0d-41be-b693-c970b18dbedc             avgFileSizeInMB
01c75278-9c0d-41be-b693-c970b18dbedc                cpu_used_avg
01c75278-9c0d-41be-b693-c970b18dbedc             dataScannedinGB
01c75278-9c0d-41be-b693-c970b18dbedc  download_workers_count_avg

metrics                               IdleTimeInHrs  avgFileSizeInMB  \
appliance_id
01c75278-9c0d-41be-b693-c970b18dbedc       1.000000         0.032743
01c75278-9c0d-41be-b693-c970b18dbedc       0.032743         1.000000
01c75278-9c0d-41be-b693-c970b18dbedc       0.022182         0.093800
01c75278-9c0d-41be-b693-c970b18dbedc       0.134145         0.558588
01c75278-9c0d-41be-b693-c970b18dbedc            NaN              NaN

metrics                               cpu_used_avg  dataScannedinGB  \
appliance_id
01c75278-9c0d-41be-b693-c970b18dbedc      0.022182         0.134145
01c75278-9c0d-41be-b693-c970b18dbedc      0.093800         0.558588
01c75278-9c0d-41be-b693-c970b18dbedc      1.000000         0.167504
01c75278-9c0d-41be-b693-c970b18dbedc      0.167504         1.000000
01c75278-9c0d-41be-b693-c970b18dbedc           NaN              NaN

metrics                               download_workers_count_avg  \
appliance_id
01c75278-9c0d-41be-b693-c970b18dbedc                         NaN
01c75278-9c0d-41be-b693-c970b18dbedc                         NaN
01c75278-9c0d-41be-b693-c970b18dbedc                         NaN
01c75278-9c0d-41be-b693-c970b18dbedc                         NaN
01c75278-9c0d-41be-b693-c970b18dbedc                         NaN

metrics                               esLatency_avg  fileDownloadTimeInHrs  \
appliance_id
01c75278-9c0d-41be-b693-c970b18dbedc       0.015515               0.337686
01c75278-9c0d-41be-b693-c970b18dbedc       0.028873               0.063530
01c75278-9c0d-41be-b693-c970b18dbedc       0.000113               0.010405
01c75278-9c0d-41be-b693-c970b18dbedc       0.007807               0.037724
01c75278-9c0d-41be-b693-c970b18dbedc            NaN                    NaN

metrics                               linkerq_avg  memory_used_max  …  \
appliance_id                                                        …
01c75278-9c0d-41be-b693-c970b18dbedc          NaN         0.323574  …
01c75278-9c0d-41be-b693-c970b18dbedc          NaN         0.362822  …
01c75278-9c0d-41be-b693-c970b18dbedc          NaN         0.596260  …
01c75278-9c0d-41be-b693-c970b18dbedc          NaN         0.598367  …
01c75278-9c0d-41be-b693-c970b18dbedc          NaN              NaN  …
```

```
metrics                               numberOfChunksScanned  \
appliance_id
01c75278-9c0d-41be-b693-c970b18dbedc               0.012446
01c75278-9c0d-41be-b693-c970b18dbedc               0.028560
01c75278-9c0d-41be-b693-c970b18dbedc               0.094911
01c75278-9c0d-41be-b693-c970b18dbedc               0.155238
01c75278-9c0d-41be-b693-c970b18dbedc                    NaN

metrics                               numberOfColsScanned  pgLatency_avg  \
appliance_id
01c75278-9c0d-41be-b693-c970b18dbedc                  NaN       0.066219
01c75278-9c0d-41be-b693-c970b18dbedc                  NaN       0.059433
01c75278-9c0d-41be-b693-c970b18dbedc                  NaN       0.022957
01c75278-9c0d-41be-b693-c970b18dbedc                  NaN       0.038979
01c75278-9c0d-41be-b693-c970b18dbedc                  NaN            NaN

metrics                               pod_cpu_usage_avg  pod_memory_usage_avg  \
appliance_id
01c75278-9c0d-41be-b693-c970b18dbedc           0.010472              0.007449
01c75278-9c0d-41be-b693-c970b18dbedc           0.048583              0.032665
01c75278-9c0d-41be-b693-c970b18dbedc           0.013278              0.022344
01c75278-9c0d-41be-b693-c970b18dbedc           0.057678              0.024285
01c75278-9c0d-41be-b693-c970b18dbedc                NaN                   NaN

metrics                               redisLatency_avg  scanTime  taskq_avg  \
appliance_id
01c75278-9c0d-41be-b693-c970b18dbedc          0.002635  0.005601   0.009926
01c75278-9c0d-41be-b693-c970b18dbedc          0.007076  0.094383   0.052507
01c75278-9c0d-41be-b693-c970b18dbedc          0.020036  0.017485   0.874397
01c75278-9c0d-41be-b693-c970b18dbedc          0.047678  0.231155   0.028069
01c75278-9c0d-41be-b693-c970b18dbedc               NaN       NaN        NaN

metrics                               tmp_taskq_avg  uniqPodCount
appliance_id
01c75278-9c0d-41be-b693-c970b18dbedc       0.012718      0.052765
01c75278-9c0d-41be-b693-c970b18dbedc       0.020660      0.007626
01c75278-9c0d-41be-b693-c970b18dbedc       0.302852      0.023543
01c75278-9c0d-41be-b693-c970b18dbedc       0.022780      0.582436
01c75278-9c0d-41be-b693-c970b18dbedc            NaN           NaN

[5 rows x 21 columns]
```

0.5 Step 4: Isolate related metrics using correlation

We now iterate over each `metric`, to see if there is any significant statistical correlation to be found across `appliance_ids`. This is done with two steps:

1. Removing outliers:
    - Remove any metrics with `3rd quantile correlation value below the cut-off`. This cut-off can be varied for depending on use cases:
        - 0.9 for Exec Dashboards
        - 0.8 for Customer Ops
        - 0.7 for L1 - support
        - 0.6 for L2 - suport

Please note that we are filtering metrics with `3rd quantile` correlation below the `moderate cut-off`. This ensures that atleast 25% of the values are correlated to reduce outliers.

2. Plot box chart to visually represent metrics with any correlation (for cutoff as 0.3).

3. The network graph indicates specific correlation edges between metrics.

0.6 Final List of metrics

The below table shows the list of `metrics` that are useful with respective correlation `cutoff`. The cut-off values can be interpreted as follows:

- below 0.3 negligible correlation
- 0.3 to 0.5 Low positive (negative) correlation
- 0.5 to 0.7 Moderate positive (negative) correlation
- 0.7 to 0.9 High positive (negative) correlation
- 0.9 to 0.1 Very High positive (negative) correlation

[4]:
```
import gravis as gv
import itertools
import networkx as nx
from IPython.display import Image

corr_vals = [0.6, 0.7, 0.8, 0.9]
mtrx_arr = []
graph_arr = []
for cutoff in corr_vals:
    arr = []
    for metr in dfc.metric.unique():
        dfcm = dfc[(dfc.metric == metr)]
        dfcm = dfcm.drop('metric', axis=1)
        dfcm = dfcm.drop(metr, axis=1)
        dfcm = dfcm.dropna(axis = 0, how = 'all')
        dfcm = dfcm.loc[:, dfcm.quantile(q=0.75) > cutoff]
        for x in dfcm.columns:
            arr.append(x)
            graph_arr.append((metr, x))
        if(cutoff == corr_vals[0]):
            if len(dfcm.columns) > 0:
                title=f'''Absolute correlation vs percent-change of {metr}
                (For median correlation greater than {cutoff})
                '''
```

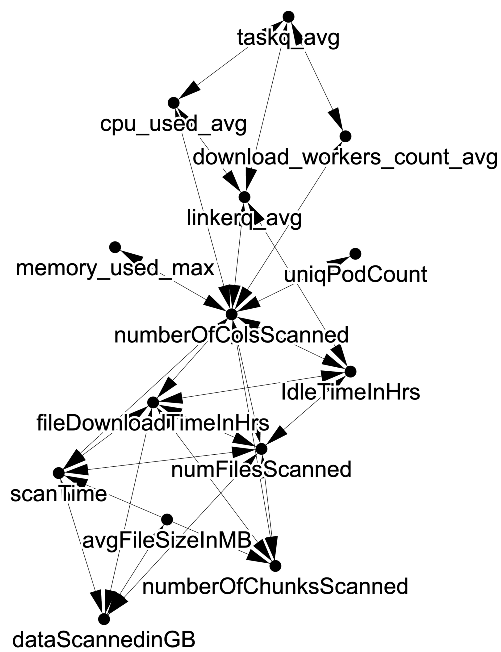
```
                dfcm.plot(kind='box',vert=False,title=title,colormap='tab20')
    arr = list(set(arr))
    arr.insert(0,cutoff)
    mtrx_arr.append(arr)

display(pd.DataFrame(list(map(list, itertools.zip_longest(*mtrx_arr,␣
 ↪fillvalue="")))))

g = nx.DiGraph()
g.add_edges_from(graph_arr)
fig = gv.d3(g
       , graph_height=800
       , use_node_size_normalization=True
       , zoom_factor=2
       , node_size_normalization_max=30
      ,use_edge_size_normalization=True
      ,use_collision_force=True
      ,node_label_size_factor=0.8
       , layout_algorithm_active=True
       )
fig.export_jpg('graph2.jpg', overwrite=True)
print("Correlation graph between appliance metrics")
Image('graph2.jpg')
# fig.display()
```

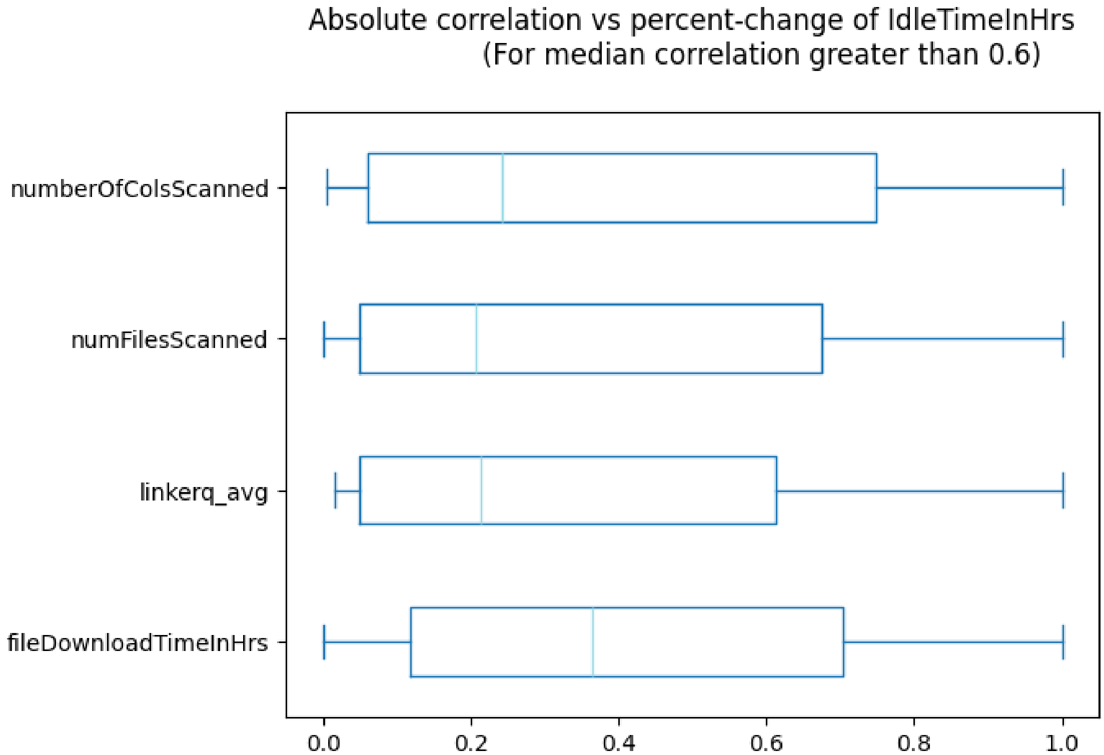
```
                           0                        1  \
0                        0.6                      0.7
1              avgFileSizeInMB          avgFileSizeInMB
2              memory_used_max          memory_used_max
3                     scanTime                 scanTime
4                  uniqPodCount          dataScannedinGB
5              dataScannedinGB             IdleTimeInHrs
6                IdleTimeInHrs  download_workers_count_avg
7   download_workers_count_avg        numberOfColsScanned
8          numberOfColsScanned             cpu_used_avg
9                 cpu_used_avg                taskq_avg
10                   taskq_avg      numberOfChunksScanned
11       numberOfChunksScanned      fileDownloadTimeInHrs
12       fileDownloadTimeInHrs              linkerq_avg
13                 linkerq_avg           numFilesScanned
14             numFilesScanned

                           2                      3
0                        0.8                    0.9
1              avgFileSizeInMB               scanTime
2                     scanTime    numberOfColsScanned
3              dataScannedinGB  numberOfChunksScanned
4   download_workers_count_avg  fileDownloadTimeInHrs
```

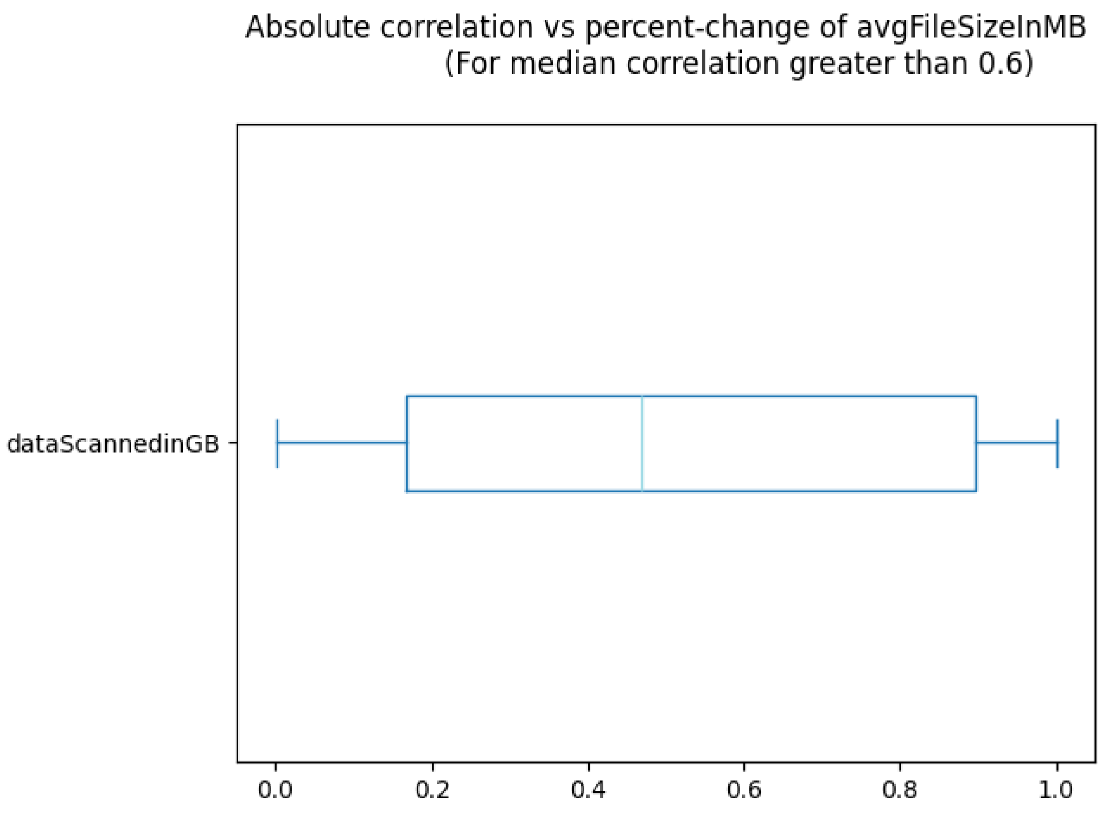
```
5           numberOfColsScanned        numFilesScanned
6         numberOfChunksScanned
7         fileDownloadTimeInHrs
8               numFilesScanned
9
10
11
12
13
14
```

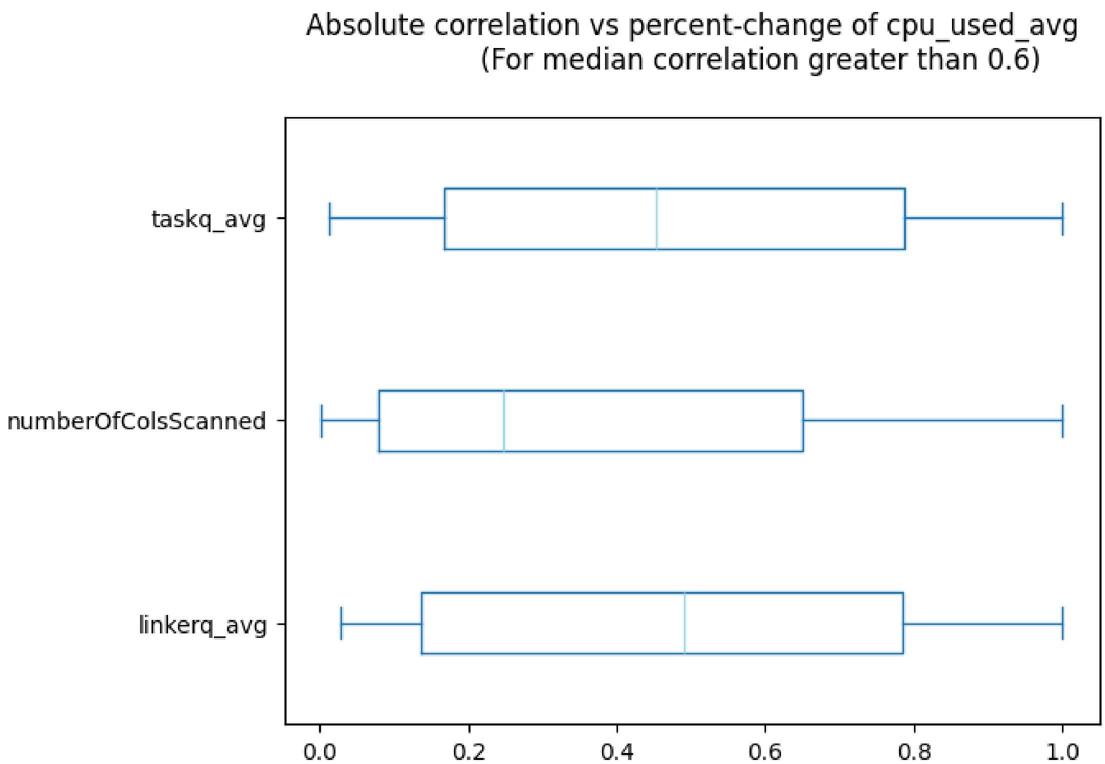
Correlation graph between appliance metrics

[4]:

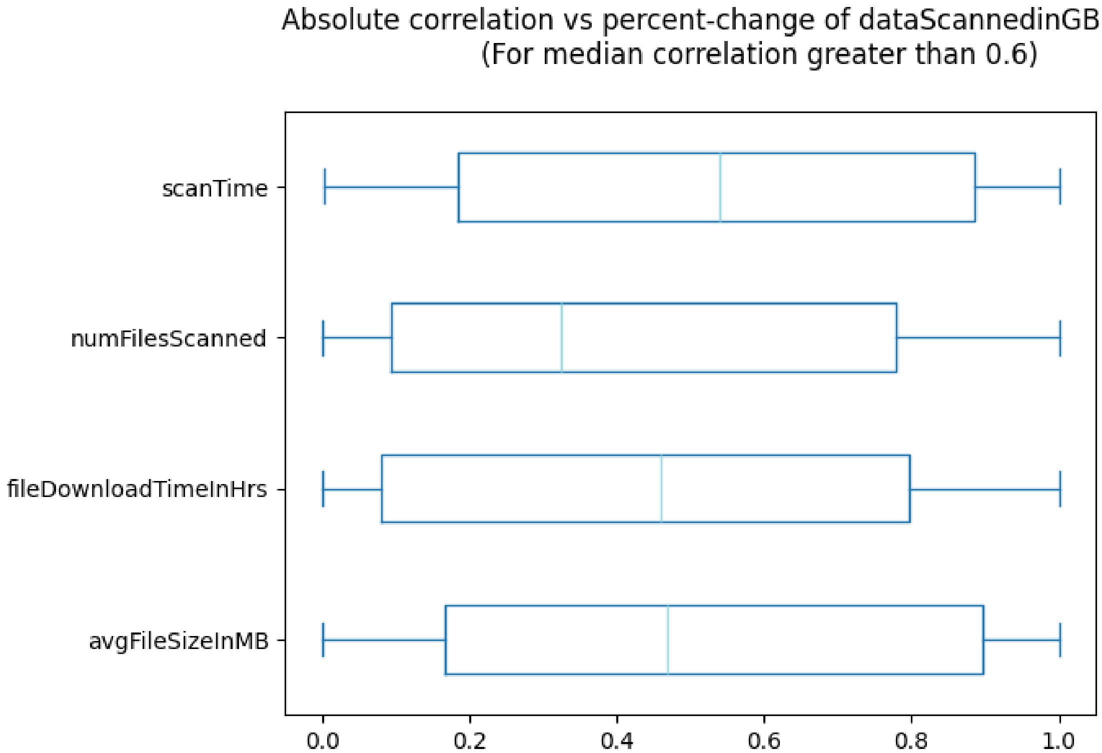


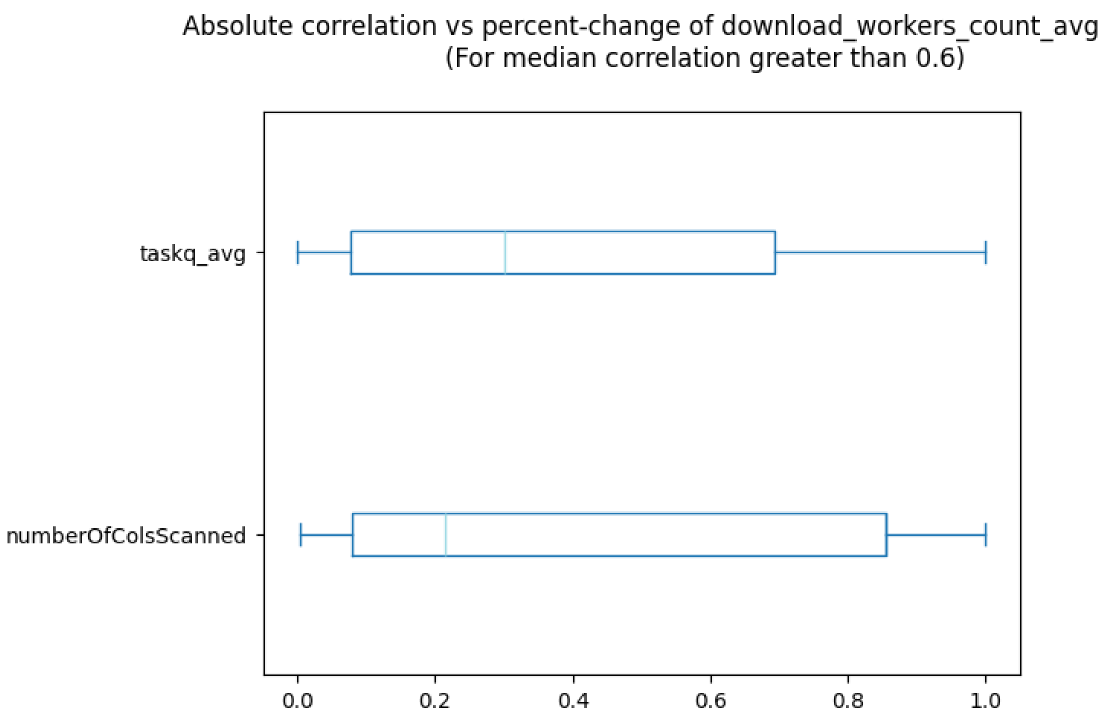

Absolute correlation vs percent-change of IdleTimeInHrs
(For median correlation greater than 0.6)
numberOfColsScanned
numFilesScanned
linkerq_avg
fileDownloadTimeInHrs
0.0
0.2
0.4
0.6
0.8
1.0

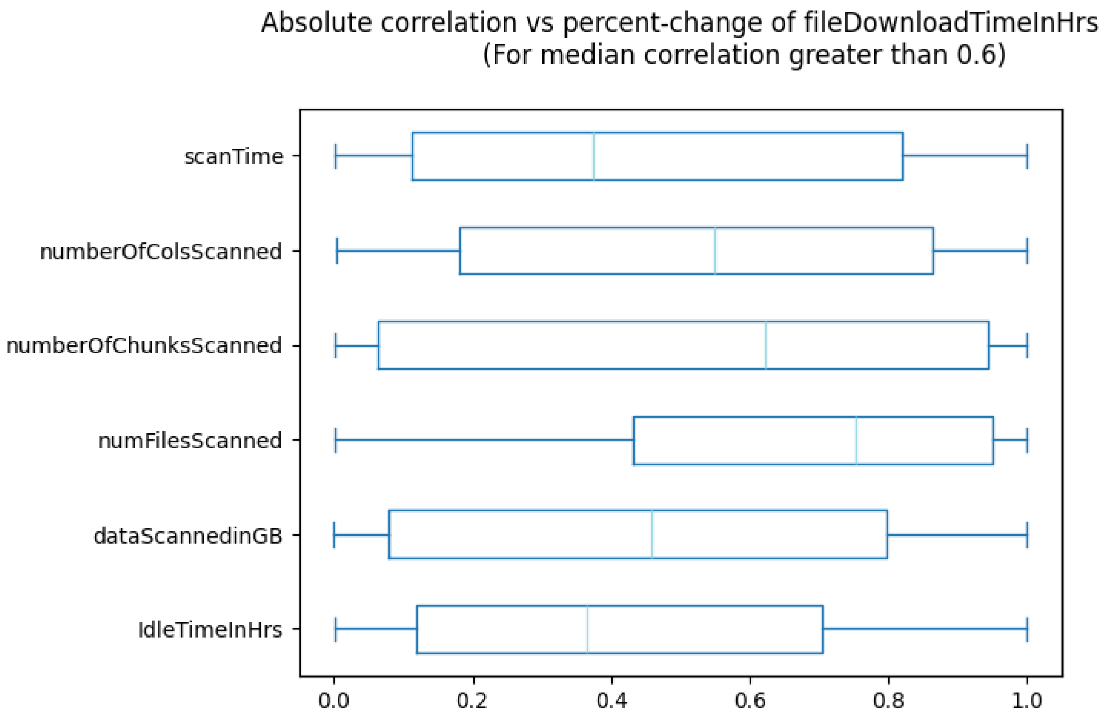

Absolute correlation vs percent-change of avgFileSizeInMB
(For median correlation greater than 0.6)
dataScannedinGB
0.0
0.2
0.4
0.6
0.8
1.0

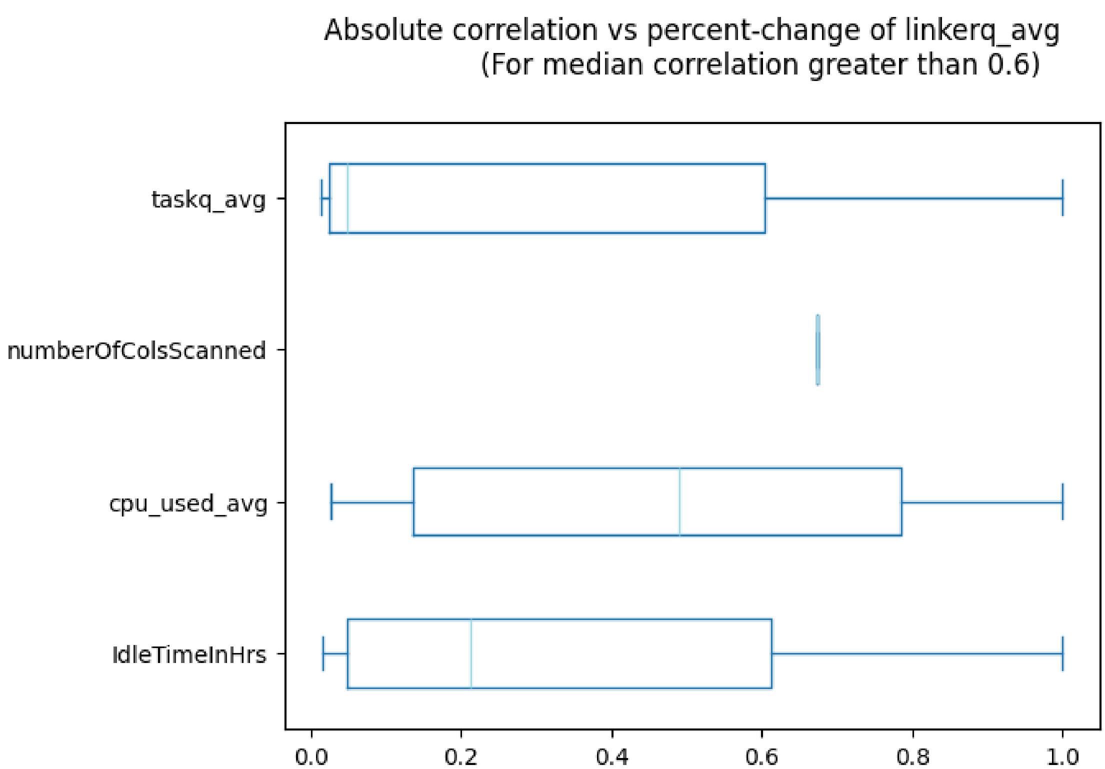


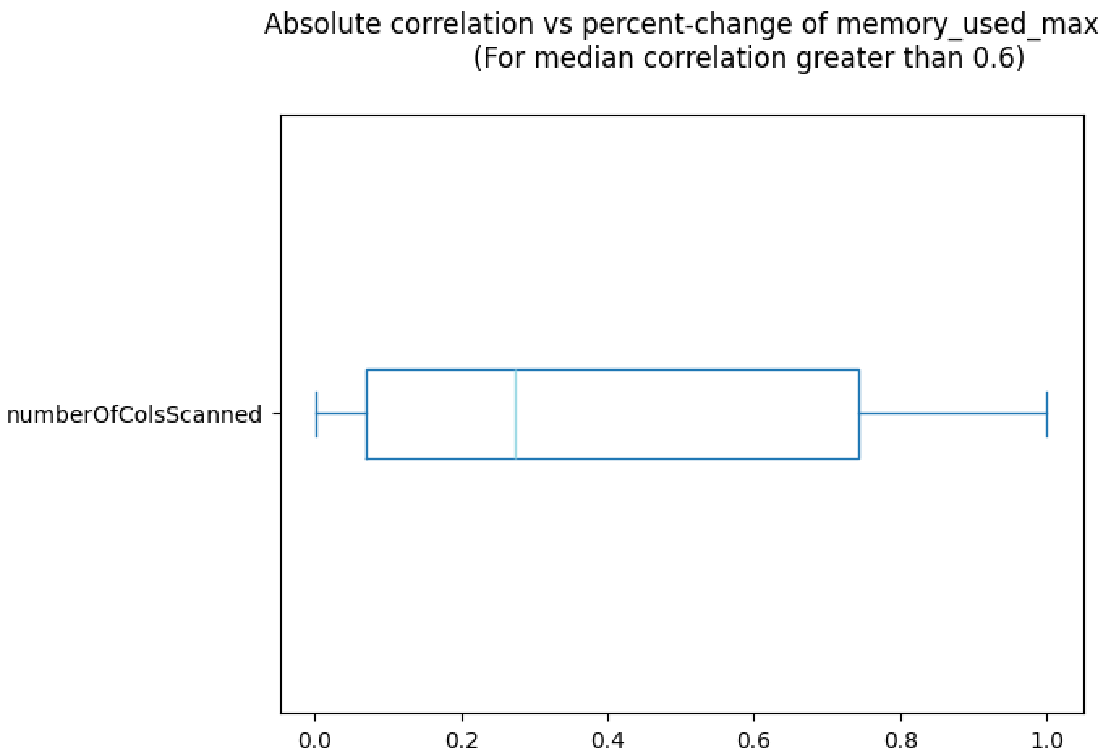
Absolute correlation vs percent-change of cpu_used_avg
(For median correlation greater than 0.6)
taskq_avg
numberOfColsScanned
linkerq_avg
0.0
0.2
0.4
0.6
0.8
1.0

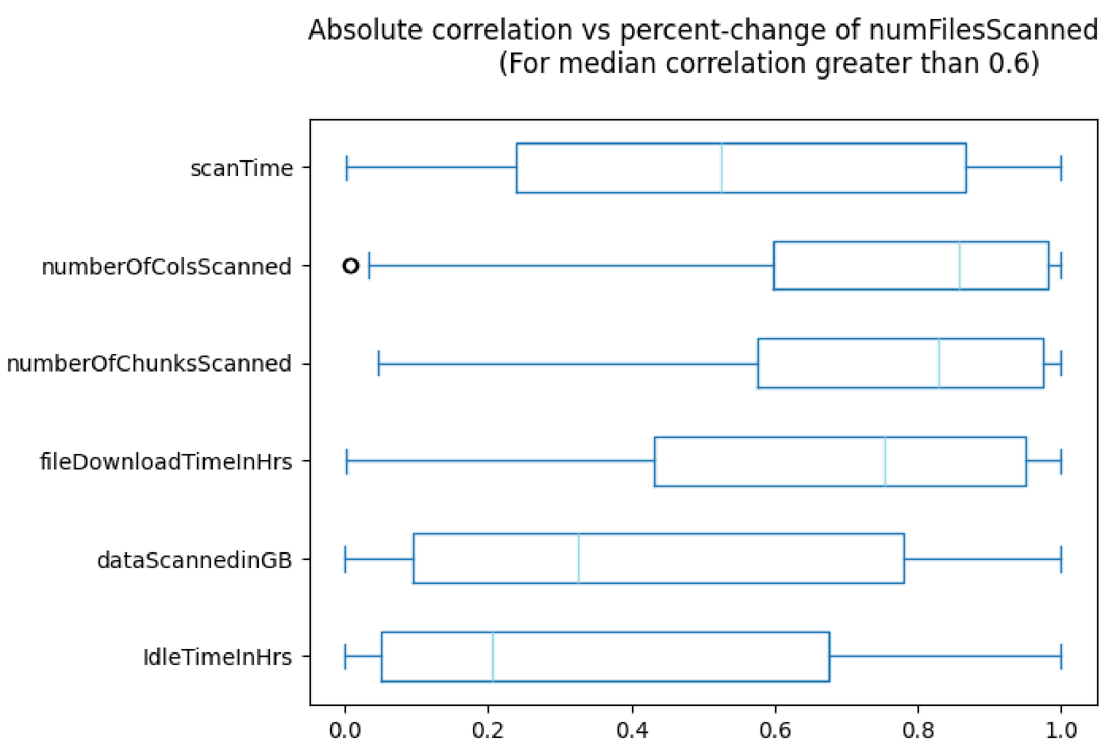

Absolute correlation vs percent-change of dataScannedinGB
(For median correlation greater than 0.6)
scanTime
numFilesScanned
fileDownloadTimeInHrs
avgFileSizeInMB
0.0
0.2
0.4
0.6
0.8
1.0


Absolute correlation vs percent-change of download_workers_count_avg
(For median correlation greater than 0.6)
taskq_avg
numberOfColsScanned
0.0
0.2
0.4
0.6
0.8
1.0

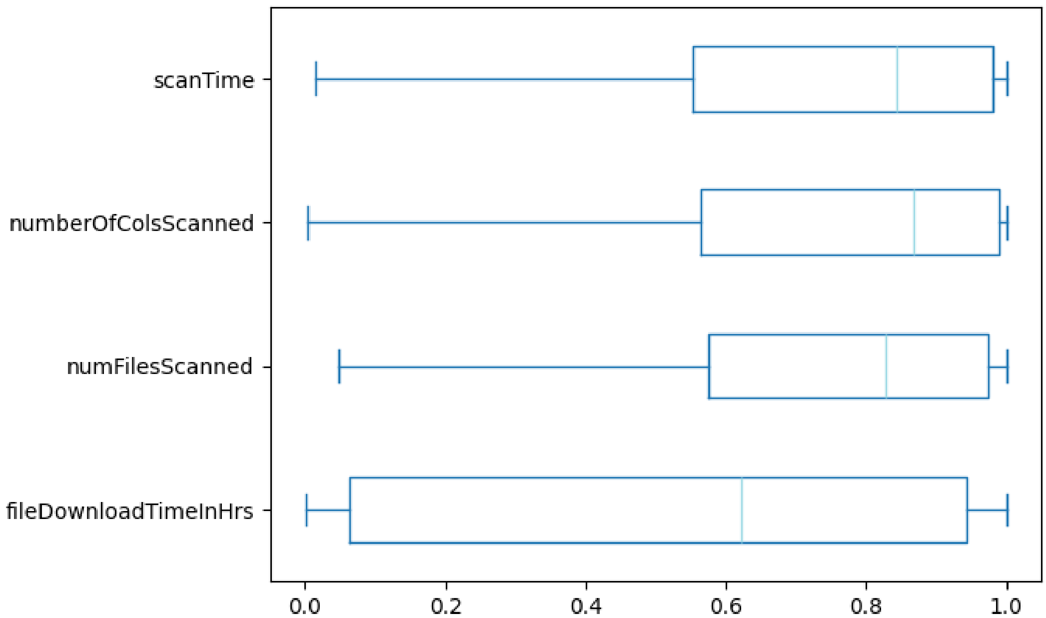

Absolute correlation vs percent-change of fileDownloadTimeInHrs
(For median correlation greater than 0.6)
scanTime
numberOfColsScanned
numberOfChunksScanned
numFilesScanned
dataScannedinGB
IdleTimeInHrs
0.0
0.2
0.4
0.6
0.8
1.0


Absolute correlation vs percent-change of linkerq_avg
(For median correlation greater than 0.6)
taskq_avg
numberOfColsScanned
cpu_used_avg
IdleTimeInHrs
0.0
0.2
0.4
0.6
0.8
1.0

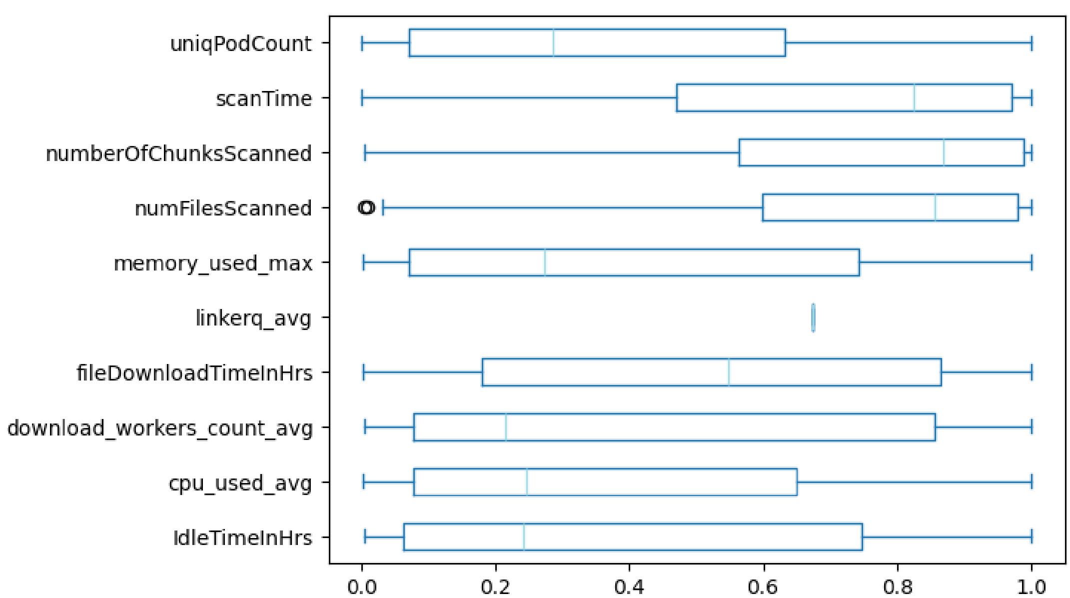

Absolute correlation vs percent-change of memory_used_max
(For median correlation greater than 0.6)
numberOfColsScanned
0.0
0.2
0.4
0.6
0.8
1.0

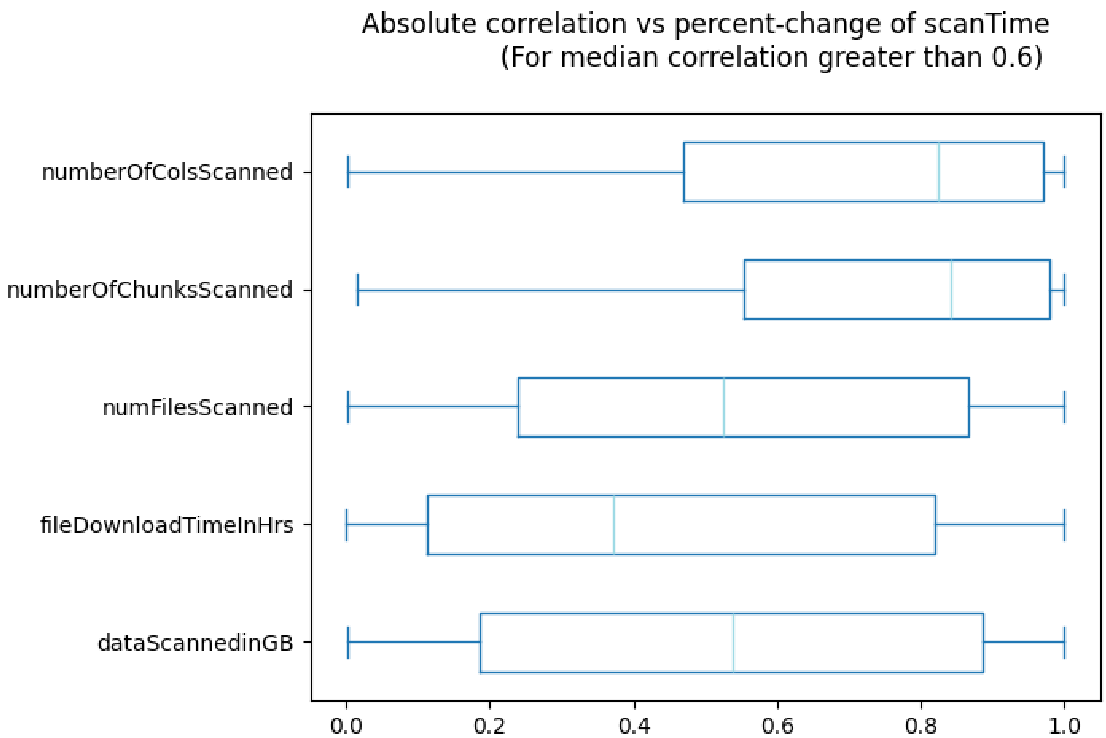

Absolute correlation vs percent-change of numFilesScanned
(For median correlation greater than 0.6)
scanTime
numberOfColsScanned
numberOfChunksScanned
fileDownloadTimeInHrs
dataScannedinGB
IdleTimeInHrs
0.0
0.2
0.4
0.6
0.8
1.0

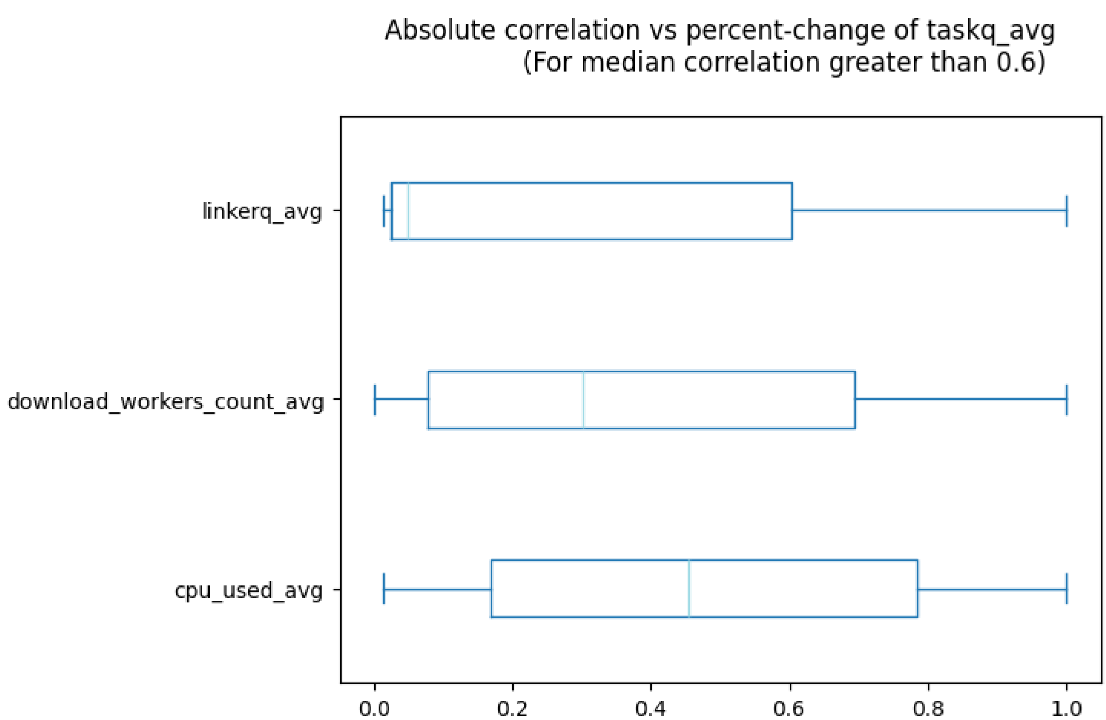
Absolute correlation vs percent-change of numberOfChunksScanned
(For median correlation greater than 0.6)

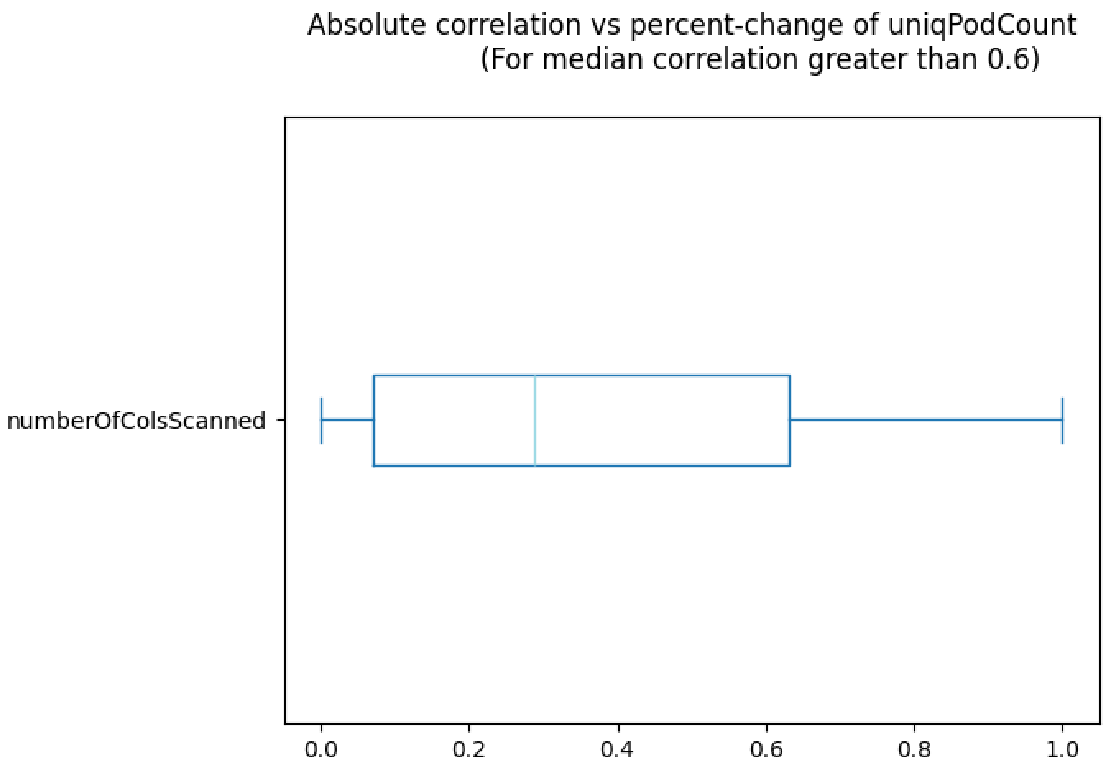


Absolute correlation vs percent-change of numberOfColsScanned
(For median correlation greater than 0.6)




Absolute correlation vs percent-change of scanTime
(For median correlation greater than 0.6)
numberOfColsScanned
numberOfChunksScanned
numFilesScanned
fileDownloadTimeInHrs
dataScannedinGB
0.0
0.2
0.4
0.6
0.8
1.0


Absolute correlation vs percent-change of taskq_avg
(For median correlation greater than 0.6)
linkerq_avg
download_workers_count_avg
cpu_used_avg
0.0
0.2
0.4
0.6
0.8
1.0


Absolute correlation vs percent-change of uniqPodCount
(For median correlation greater than 0.6)
numberOfColsScanned
0.0
0.2
0.4
0.6
0.8
1.0