

## Timing Studies for SAM declare Metadata command

Tests to measure the capabilities of a v7\_3\_0 SAM dbserver were performed. These tests are documented in this note. This testing occurred on thefcdflnx3.fnal.gov. To perform these tests, a python script was run. The script used is in appendix A. These jobs used the SAM DB server user\_int (v7\_3\_0) running on cdfsam05.fnal.gov. Table 1 lists the number of sections, type of test, start and stop times

**Table 1 List of SAM getMetadata tests performed**

	Type of test	# of job sections	Submit time	End time
1	Concurrent declare Metadata	10-50 jobs	Tue Sep 6 00:18:00 2005	Tue Sep 6 02:04:00 2005

The purpose of these tests was to show the performance of the dbserver to concurrent declare metadata commands.

The average and RMS for the declare metadata execution time as a function of the number concurrent commands.

9 cmds mean = 19.3256 RMS = 2.92697

10 cmds mean = 16.9128 RMS = 2.50036

19 cmds mean = 39.301 RMS = 5.75939

20 cmds mean = 39.3875 RMS = 5.64848

26 cmds mean = 76.0554 RMS = 7.78279

28 cmds mean = 65.671 RMS = 7.56844

29 cmds mean = 57.7923 RMS = 8.85945

30 cmds mean = 66.2936 RMS = 8.01189

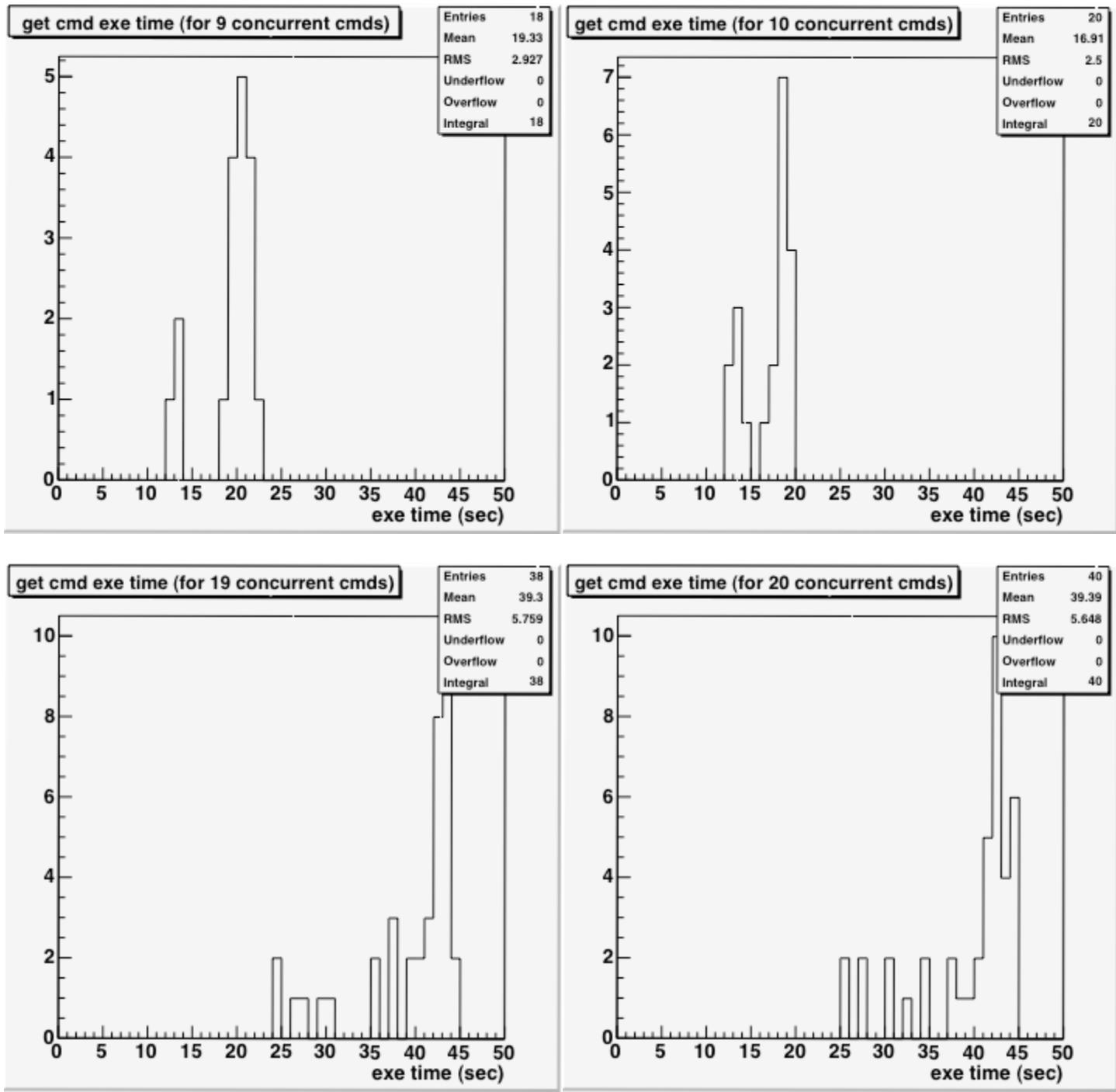
46 cmds mean = 112.185 RMS = 11.7596

47 cmds mean = 115.753 RMS = 12.0578

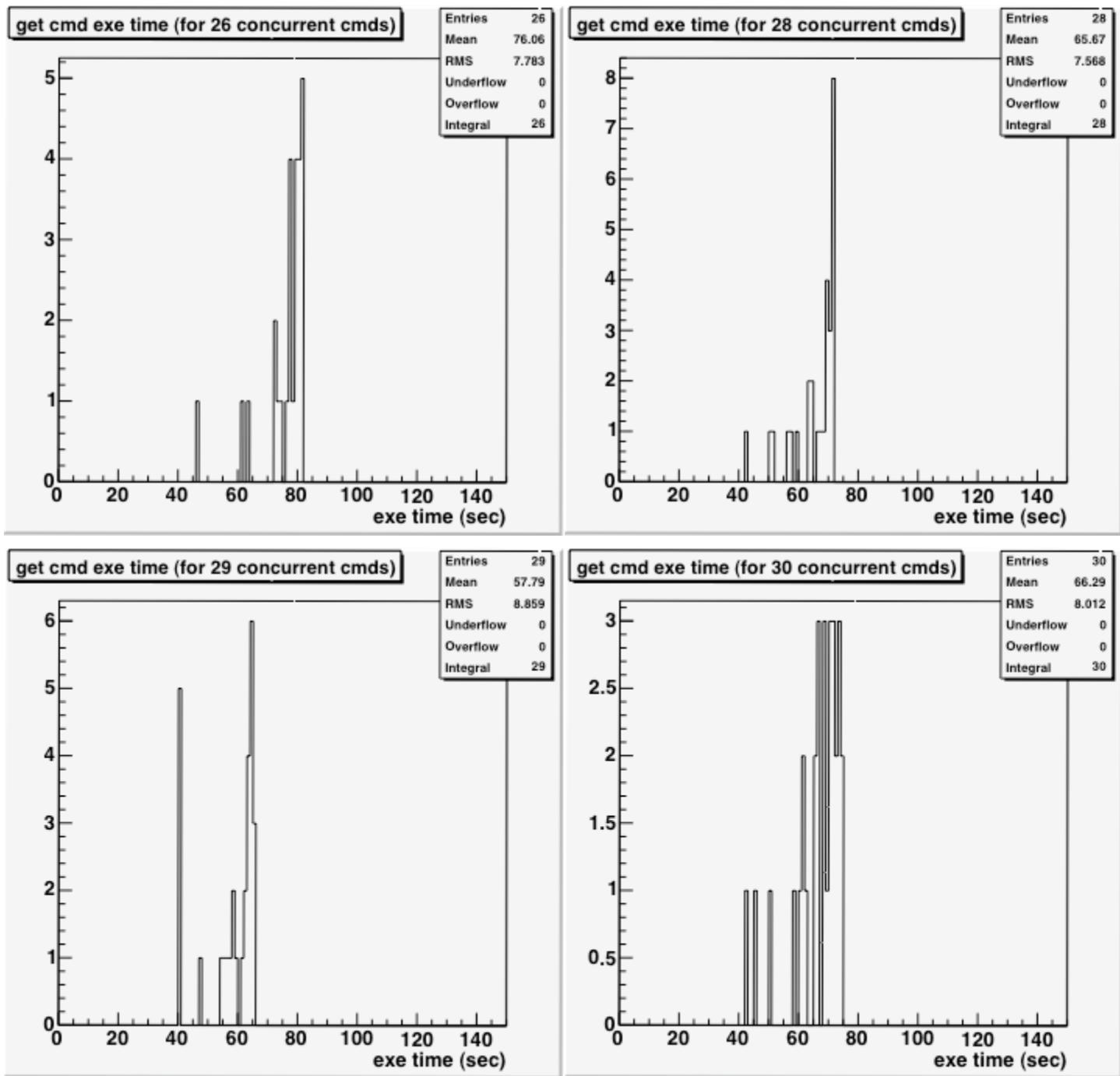
48 cmds mean = 114.41 RMS = 10.752

49 cmds mean = 106.656 RMS = 10.2887

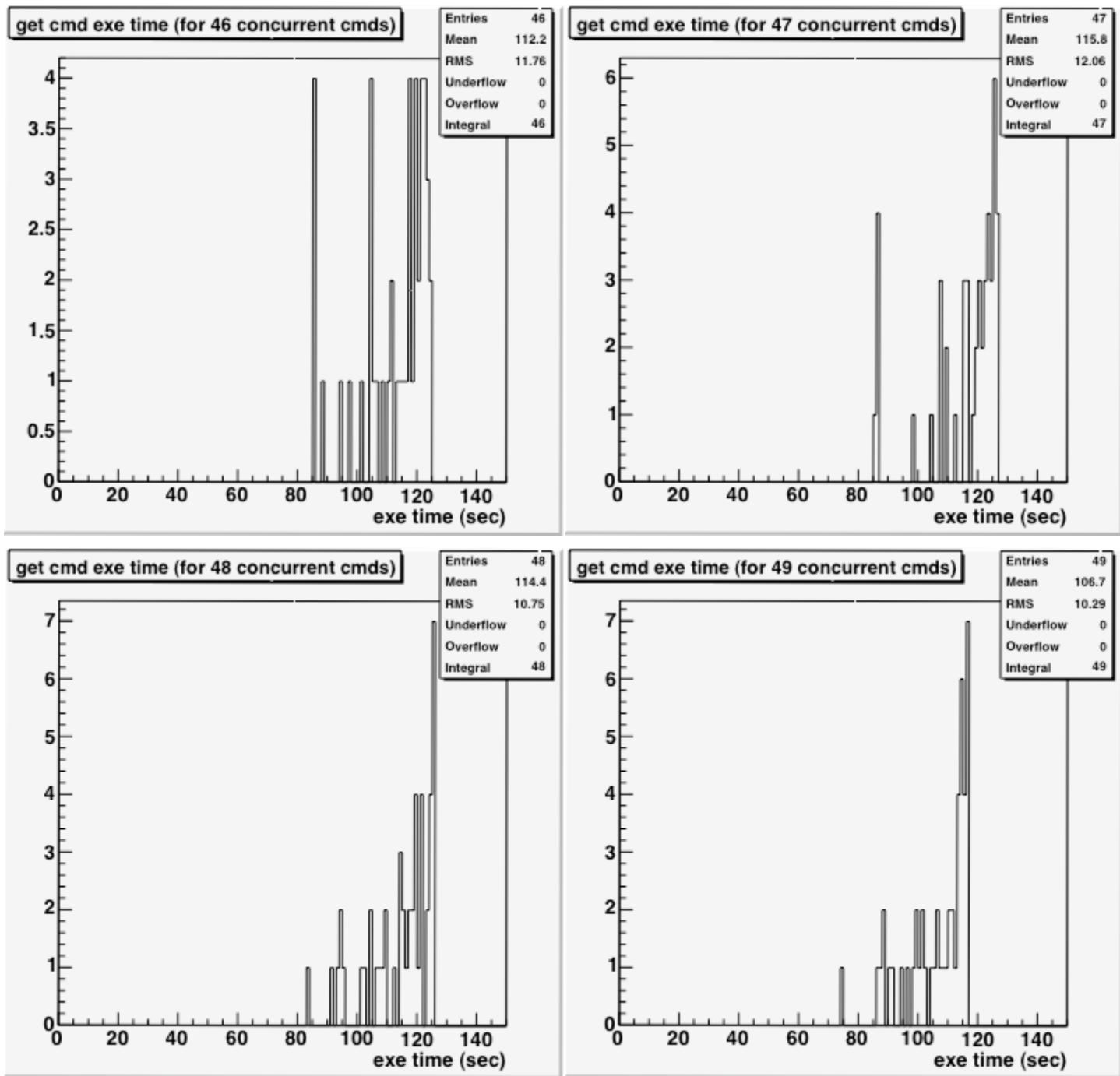
The next twelve plots show the distribution length of time each command took.



**Figure 1 - Distribution of command execution time for declare meta data command as a function of the number of concurrent commands**



**Figure 2 - Distribution of command execution time for declare meta data command as a function of the number of concurrent commands**



**Figure 3 - Distribution of command execution time for declare meta data command as a function of the number of concurrent commands**

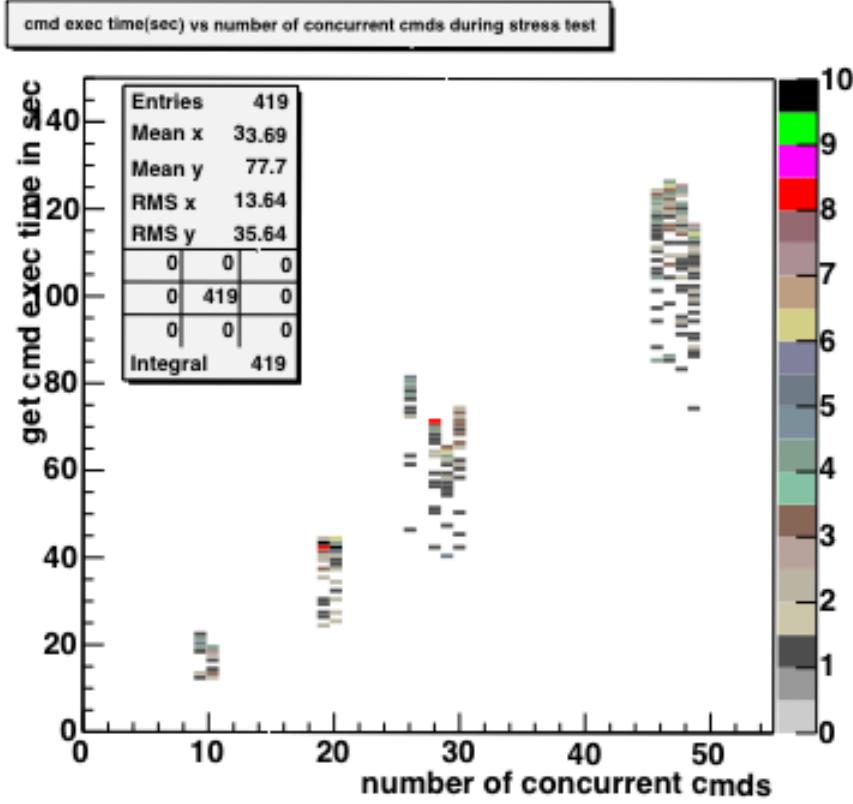


Figure 4 Command execution time vs number of concurrent commands. The histograms in figures 1-3 are equivalent to slices of this plot.

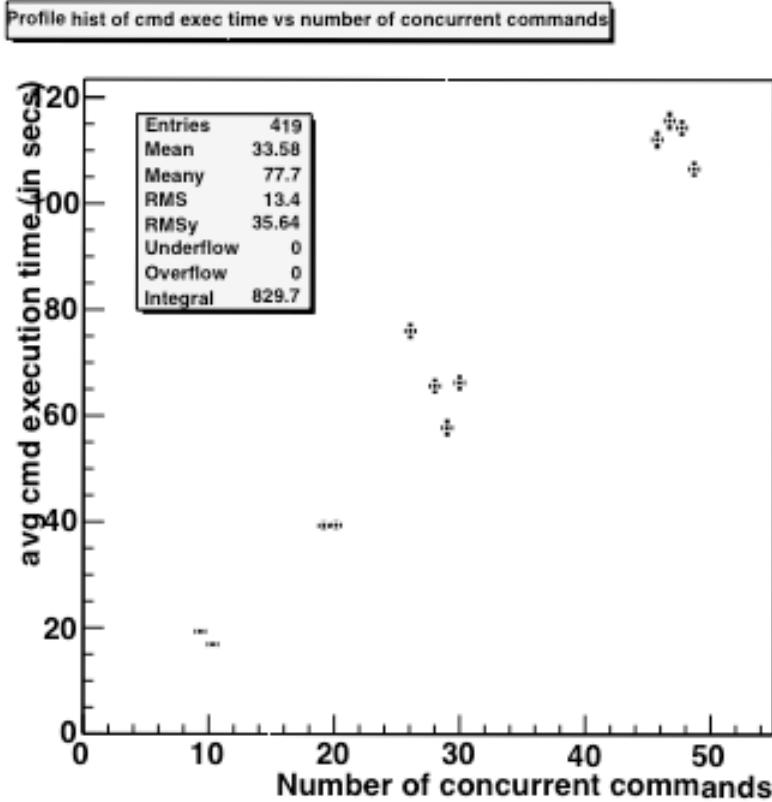
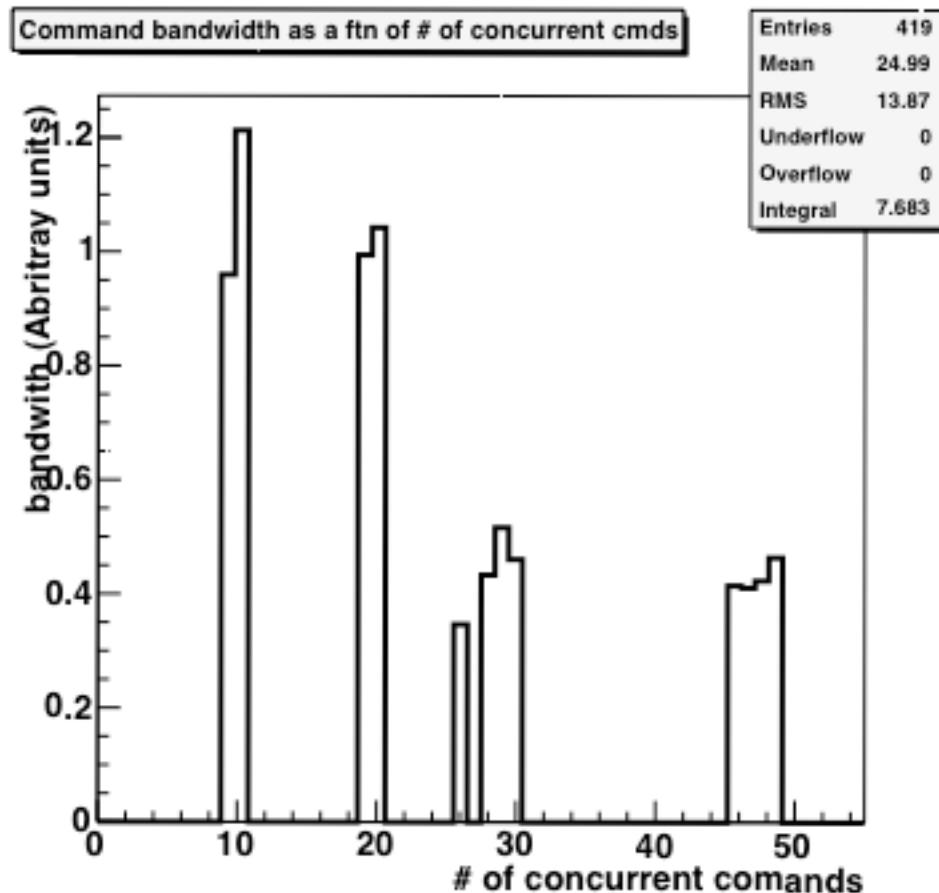


Figure 5 Profile histogram of the command execution time vs number of concurrent commands. The command execution time grows as a function of the number of concurrent declare metadata commands.

In order to determine if the growth in command execution time results in a degradation of the performance of the db server. The command “bandwidth” was determined. I define the command bandwidth as the sum of the reciprocal of the command execution times for a given test. If there was no degradation in the performance of the db server under load then one would expect the histogram in figure 6 to be flat instead of showing a falling spectrum.



**Figure 6 Command "bandwidth" of the declare metadata command as a function of the number of concurrent commands.**

## Conclusions:

Given the distinct possibility that CDF can expect to have several declare meta commands occur concurrently and it takes on average > 15 seconds per command to declare the limited amount of metadata for this test. Work should be done to try to speed up the db server/ Oracle performance further. Additional tests should include measure the declare metadata timing with the same amount of data that CDF uses for the intermediate files created during B dataset skimming or the Production farm processing.

## Appendex A: Python source code used in test.

```
#!/usr/bin/env sampy
#
#      declare-metadata.py
#
#      script to do a declare of dummy metadata many times as a test.
#
import string
import sys
import os
import getopt
import commands
import time
import SAM
from Sam import sam
from SamFile.SamDataFile import SamDataFile
from SamException import SamExceptions
from SamStruct.SamBoolean import SamBoolean
from SamStruct.DbServantConnectionInfoList_v2 import
DbServantConnectionInfoList_v2

import stat
from time import gmtime, strftime, localtime
from threading import Thread
class testit(Thread):
    def __init__(self,filenum):
        Thread.__init__(self)
        self.filenum=filenum
        self.status=''
    def run(self):
        #
        # Establish create the bulk of the dummy metadata
=====
        #

        # build metadata
        metadata = {}
        metadata['fileType'] = SAM.DataFileType_PhysicsGeneric
        metadata['fileSize'] = '99999KB'
        metadata['fileContentStatus'] = 'good'
        metadata['group'] = 'cdf'

        metadata['params'] = {}
        # global
        metadata['params']['global'] = {}
```

```

metadata[ 'params' ][ 'global' ][ 'description' ] = 'This is a dummy
description to fill space'
# cdf
metadata[ 'params' ][ 'cdf' ] = {}
metadata[ 'params' ][ 'cdf' ][ 'dataSet' ] = 'metadata-declare-test'
metadata[ 'params' ][ 'cdf' ][ 'html' ] = None
metadata[ 'params' ][ 'cdf' ][ 'analysis_group' ] = 'test'

metadata[ 'dataTier' ] = 'unknown'
metadata[ 'firstEvent' ] = '1'
metadata[ 'lastEvent' ] = '9999'
metadata[ 'eventCount' ] = '9999'
metadata[ 'startTime' ] = '%s' % strftime("%d-%b-%Y", gmtime())
metadata[ 'endTime' ] = '%s' % strftime("%d-%b-%Y", gmtime())
metadata[ 'appFamily' ] = 'file-import'
metadata[ 'appName' ] = 'file-import'
metadata[ 'appVersion' ] = '1.00'

metadata[ 'datastream' ] = 'unidentified'

#      create the dummy file name
timestamp = strftime("_%Y%m%d%H%M%S", gmtime())
filename = 'test_' + str(self.filenum) + timestamp
metadata[ 'fileName' ] = filename

print 'filenum = %d filename = %s' %(self.filenum,filename)

now = time.time()
min = int(strftime("%M",localtime(now)))
sec = int(strftime("%S",localtime(now)))
    nmin = min%10
tens = min/10
    shift_min = 13 + tens*10
if nmin < 5:
    shift_min = 7 + tens*10

sleep_sec = (60-sec) + 60*(shift_min-min)
print ' now = %s sleep_sec = %d'
%(strftime("%H:%M:%S",localtime(now)),sleep_sec)
time.sleep(sleep_sec)

task_start=time.time()
starttime = time.time()
fileId = samdeclareFile(metadata=metadata)
stoptime=time.time()
elapsed_time = stoptime-starttime
starttime_string = '%s' %strftime("%Y%m%d
%H%M%S",localtime(task_start))
#      print 'declareFile %s %.3f' %(starttime_string,elapsed_time)

```

```

        self.status = 'declareFile %s %.3f'
%(starttime_string,elapsed_time)

#
# first check that sam environment has been setup
#
job_start=time.time()
job_starttime=time.clock()
job_start_string = '%s' % strftime("%Y-%m-%d %H:%M:%S",
localtime(job_start))

print 'Job declaring started: %s  % (job_start_string)

try:
    sam2 = os.environ['SETUP_SAM']
except:
    print "Error: sam not set. 'setup sam' before running me"
    sys.exit(1)

dbservername = os.environ['SAM_DB_SERVER_NAME']

job_elapsed_time = 0

#sam_station=os.environ['SAM_STATION']
#sam_project=os.environ['SAM_PROJECT']
#sam_dataset=os.environ['SAM_DATASET']

#
# parse the options
#
file_limit=50

try:
    optlist, args = getopt.getopt(sys.argv[1:], 'f', ['file_limit='])
except getopt.GetoptError, e:
    sys.exit(1)

for key, value in optlist:
    if key == '--file_limit':
        file_limit=long(value)

print 'Number of metadata declares = %d' %(file_limit)

filelist = []
upper_limit = file_limit + 1
for filenum in range(1,upper_limit):

```

```
current = testit(filenum)
filelist.append(current)
current.start()

for files in filelist:
    files.join()
    print files.status

#
#  get the total elapsed time etc.
#
job_stop=time.time()
job_stoptime=time.time()

job_stoptime_string = '%s' %strftim("-%Y-%m-%d
%H:%M:%S",localtime(job_stop))
job_elapsed_time = job_stop-job_start

print 'declare-metadata finished at %s - it took %.3f secs to run python
script ' %(job_stoptime_string,job_elapsed_time)
```

