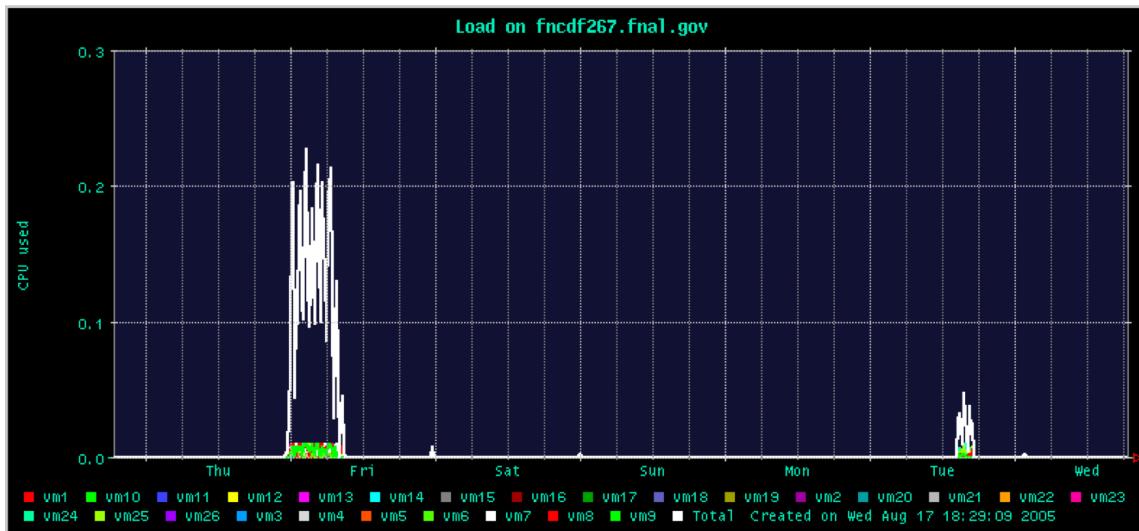


# Testing of SAM metadata declaration

This note describes the testing of the SAM metadata declaration. The python script used for the testing can be found in appendix A. The client used was version 7\_2\_0, the user integration sam db server was used v7\_3\_0 running on cdfsam05.fnal.gov. The caf-int SAM station (version v6\_0\_2\_1 on cdfsam09.fnal.gov).

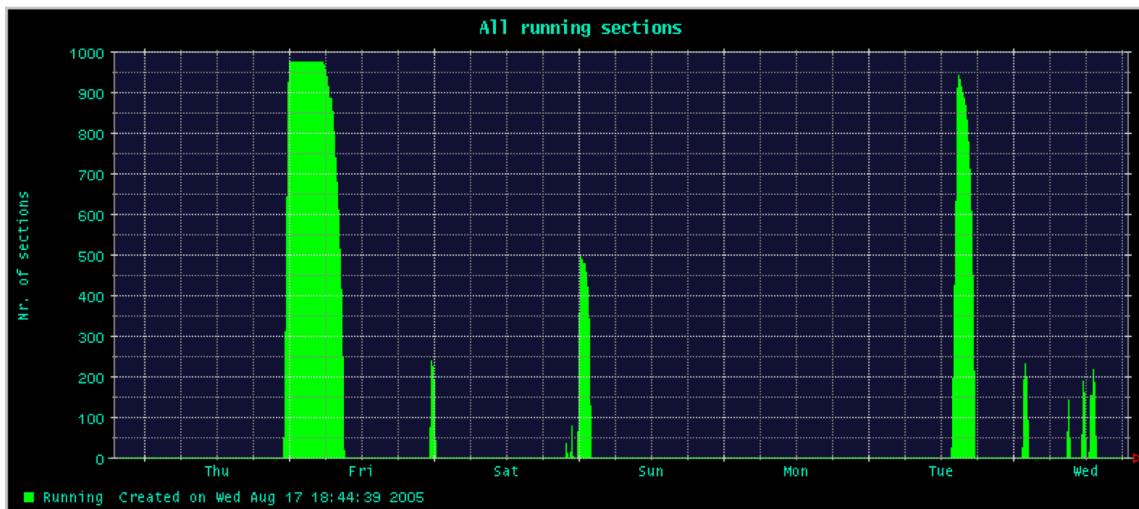
The test jobs were submitted to the fncdf CAF (running 1000 Virtual Machines on ~ 30 PC's). The load on the PC's is low as is shown in figure 1.



**Figure 1: CPU useage of CAF worker node during testing of SAM metadata declaration**

## Testing methodology:

Between 1 and 10 simultaneous CAF jobs were submitted each job with 50 sections. Each job section declared the metadata for 41 files with a 15 sleep between each metadata declaration. The time of each metadata declaration was recorded along with the length of time it took to complete the command. The testing took place on August 12-14 2005 and August 17 2005. Figure shows the number of sections used by these jobs:



**Figure 2: Number of running sections during declare metadata testing**

## Test Results for one job running simultaneously with 50 sections:

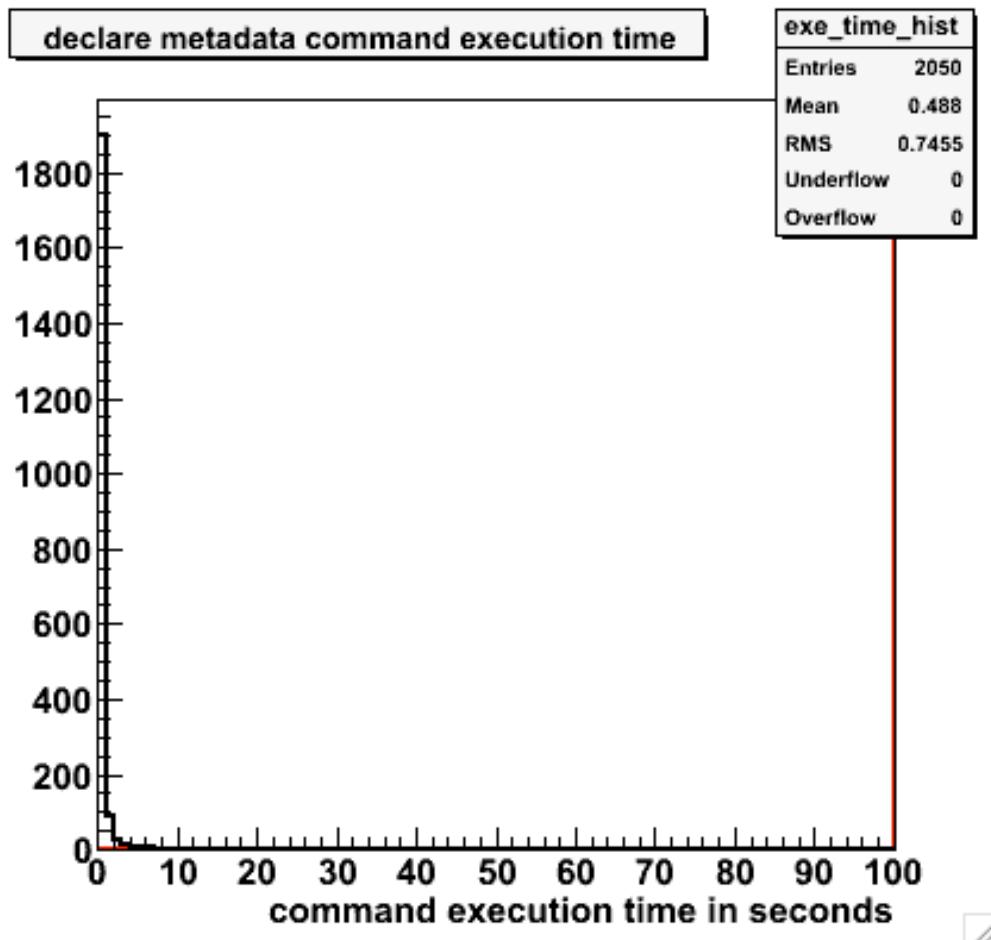


Figure 3: Execution time of declare metadata command for 1 simultaneous jobs with 50 sections.

### Command bandwidth as a ftn of time during test

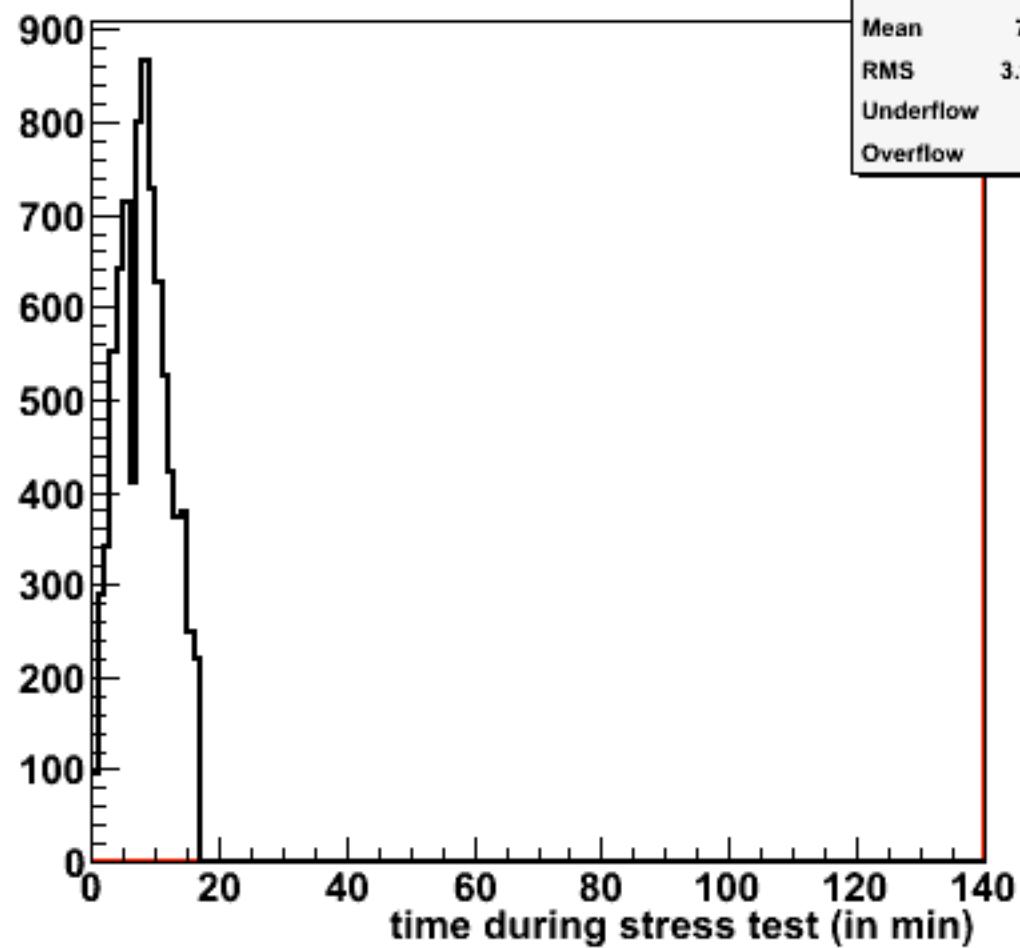


Figure 4: Command bandwidth is defined as number of commands per minute weighted by 1/ command duration – for 1 simultaneous job w/ 50 sections

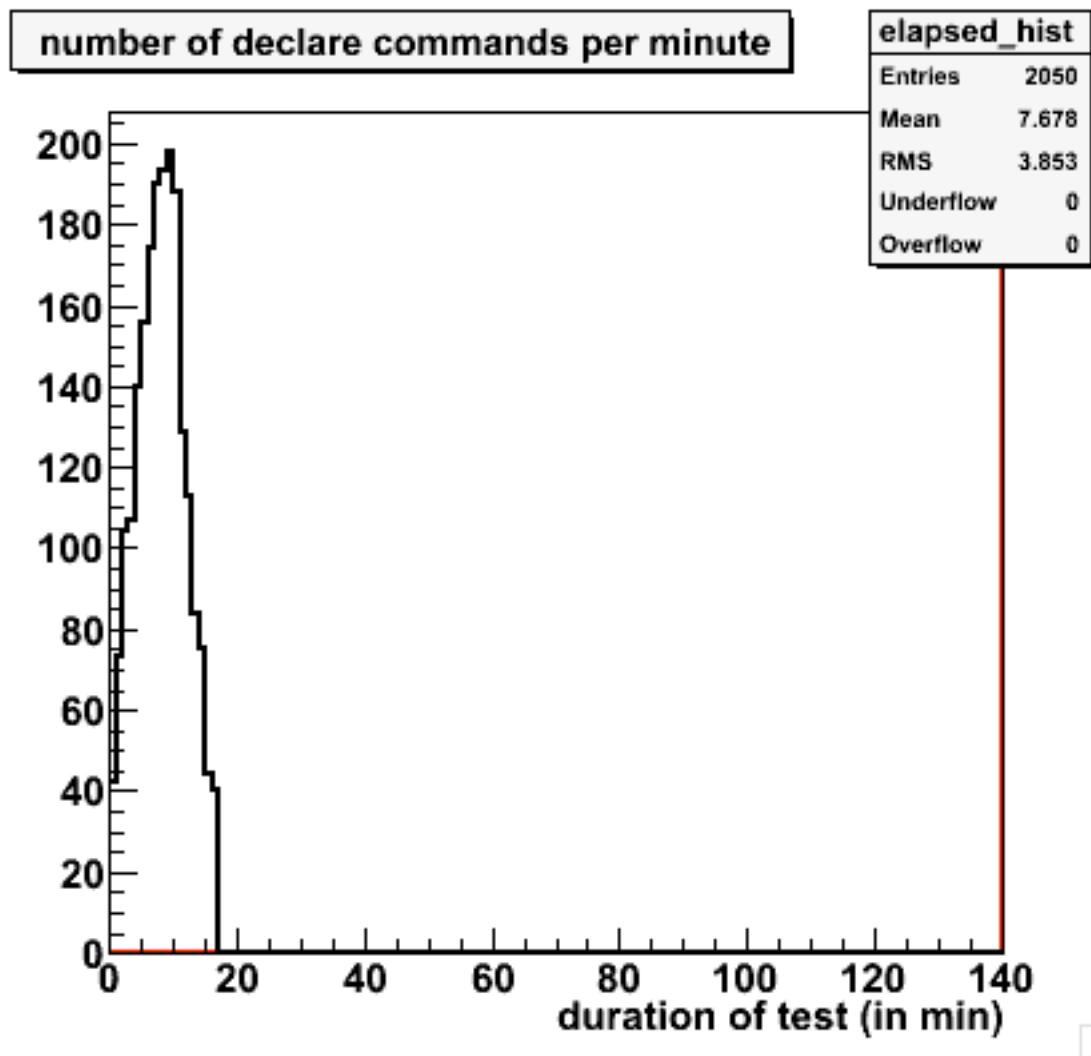


Figure 5 Number of declare commands per minute for 1 simultaneous job with 50 sections

Profile hist of cmd exec time vs cmd time during stress test

hprof	
Entries	2050
Mean	8.178
Meany	0.488
RMS	3.853
RMSy	0.7455
Underflow	0
Overflow	0

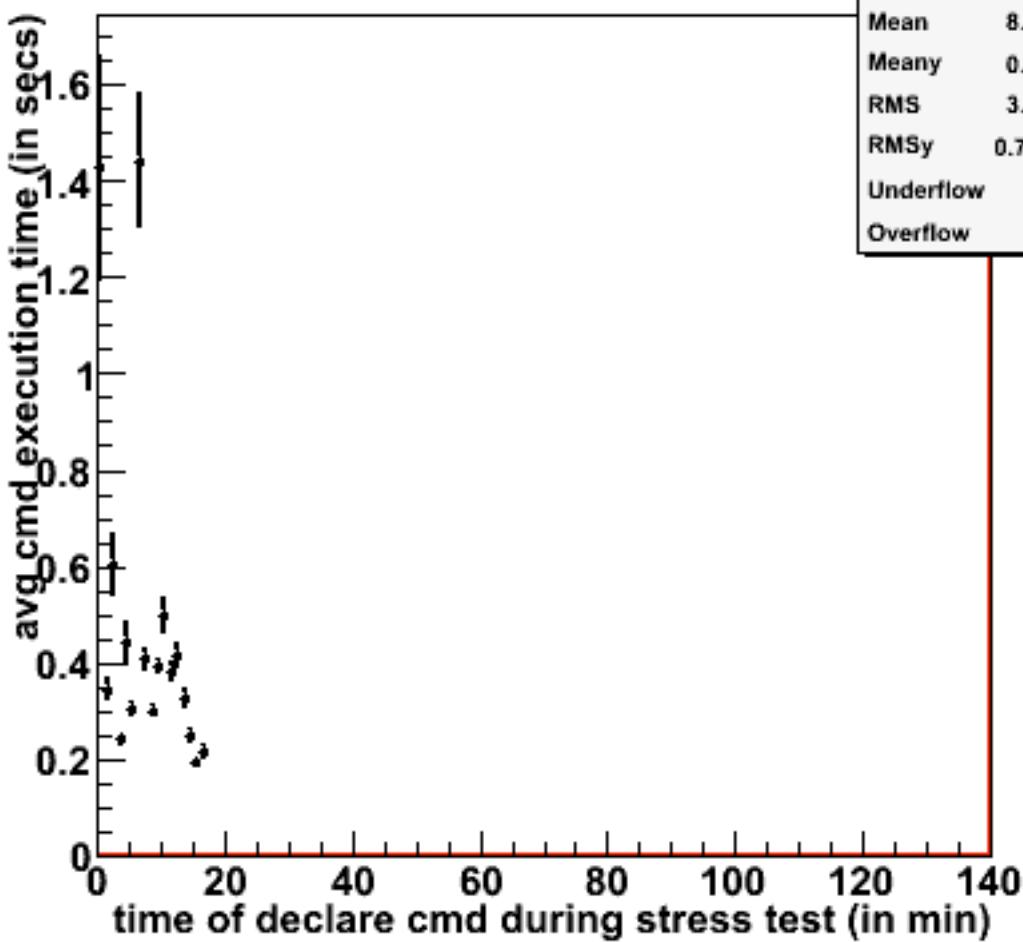
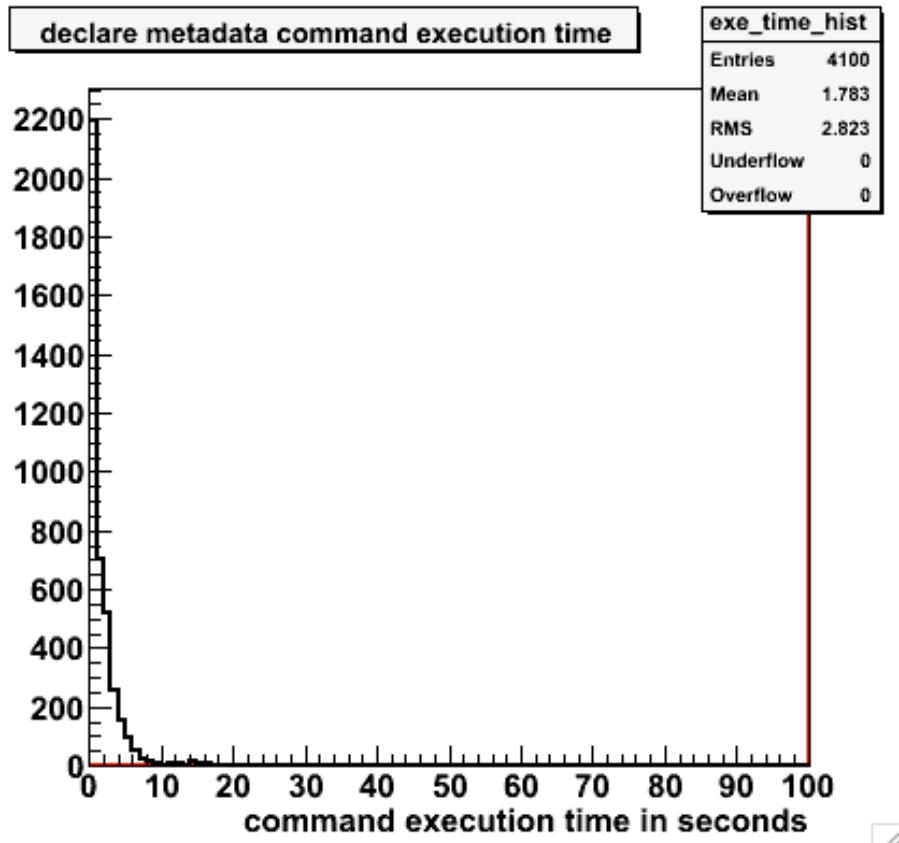


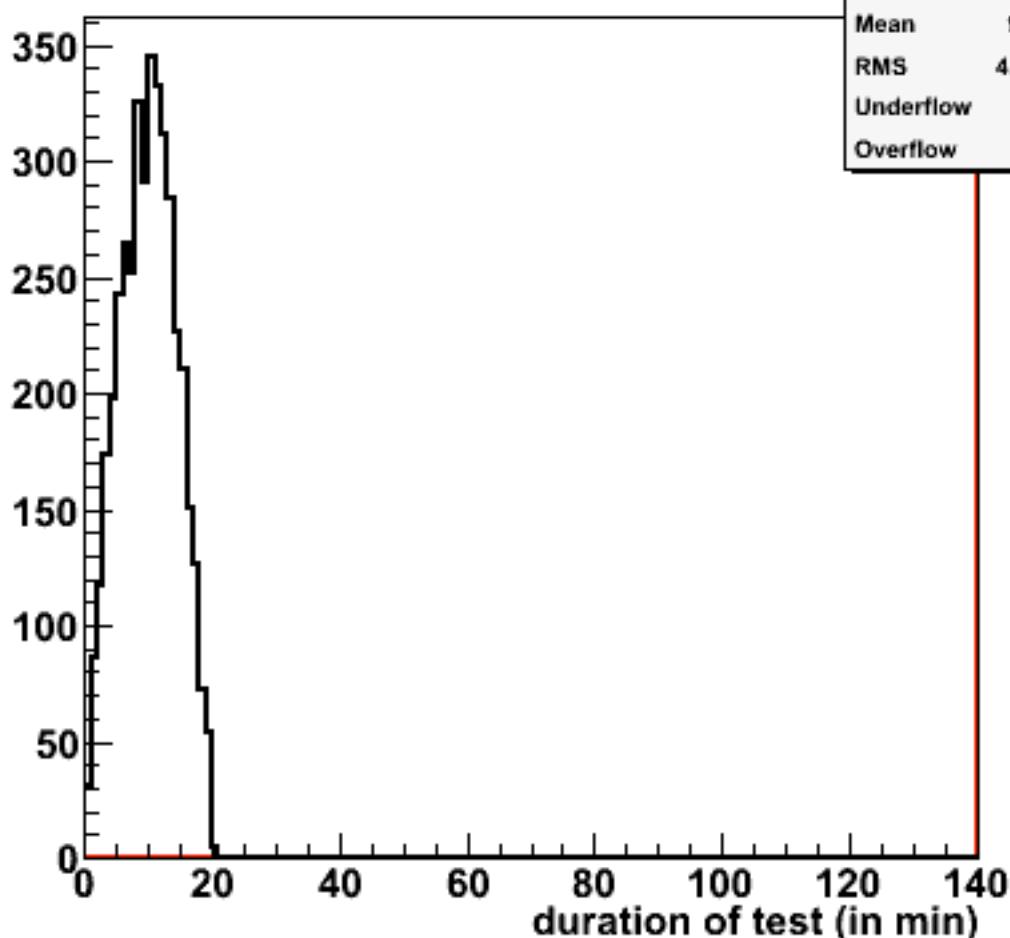
Figure 6: Profile histogram of avg command execution time as a ftn of time during test for 1 simultaneous job with 50 sections

## Test Results for: Two simultaneous jobs w/ 50 sections each:

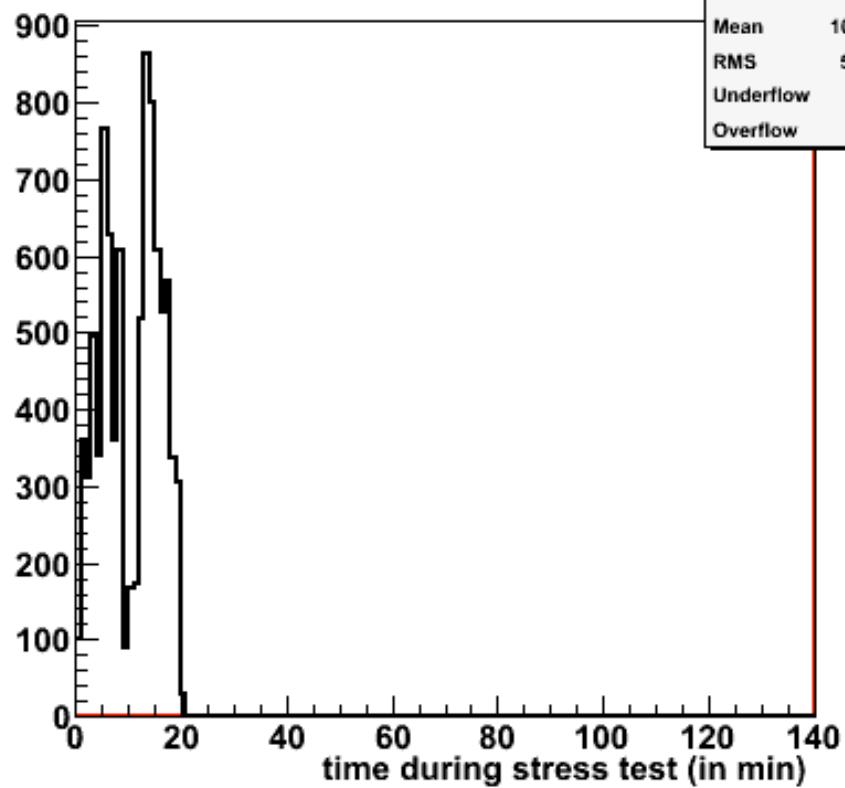


### number of declare commands per minute

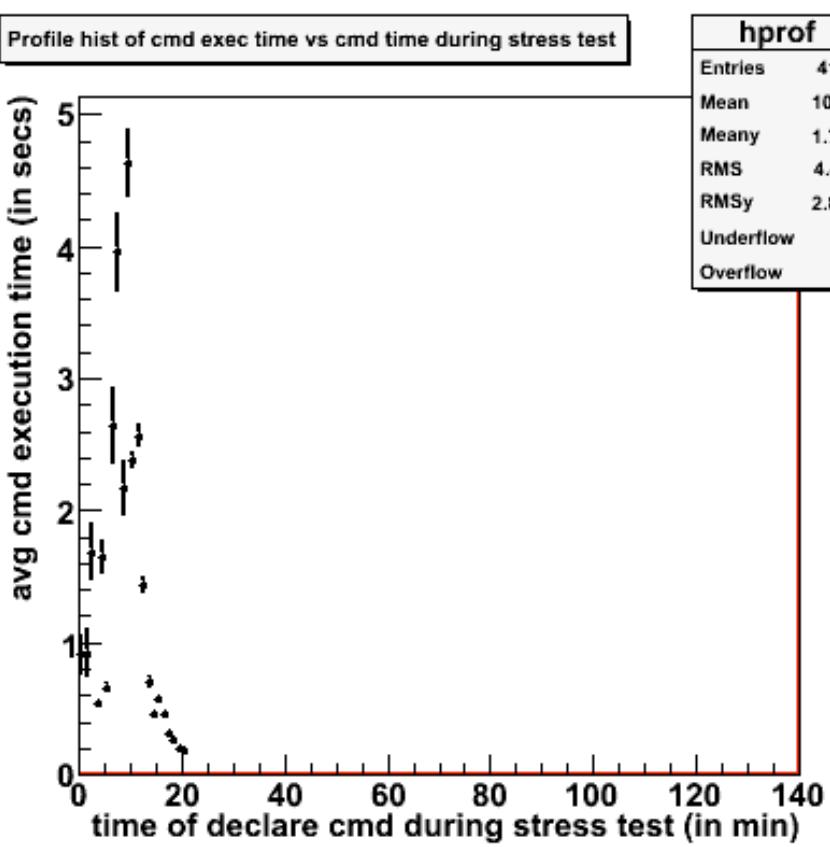
elapsed_hist	
Entries	4100
Mean	9.58
RMS	4.441
Underflow	0
Overflow	0



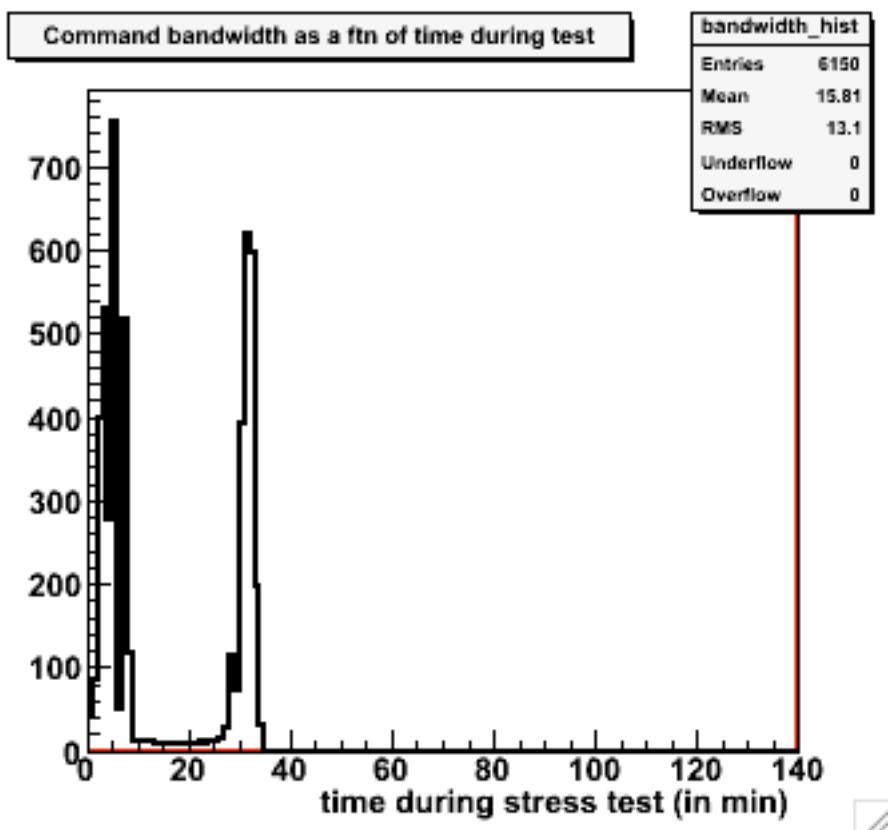
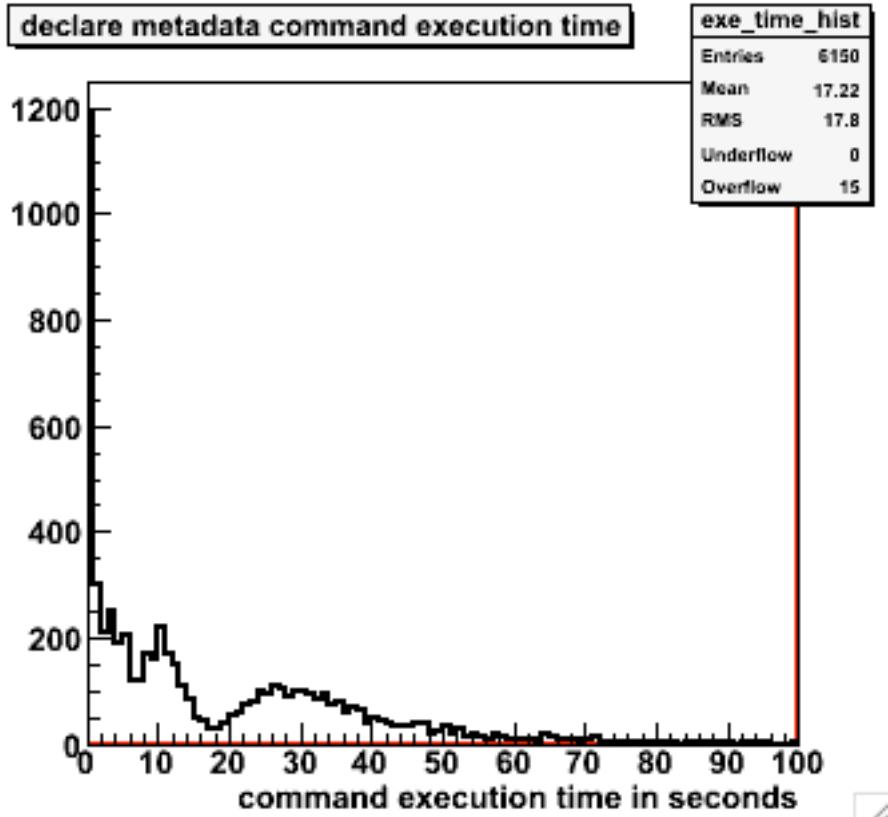
Command bandwidth as a fn of time during test



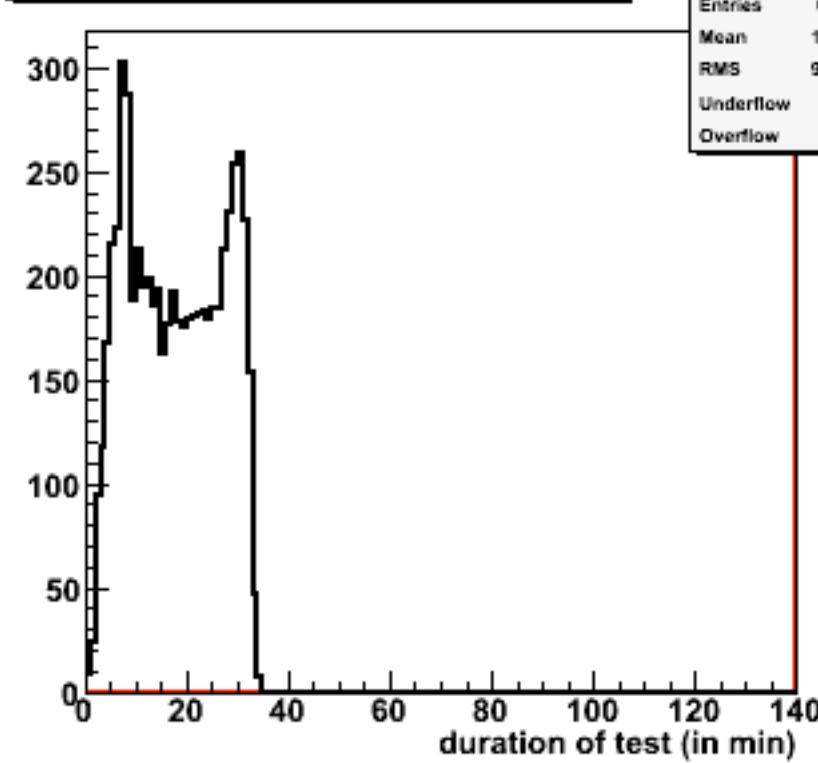
Profile hist of cmd exec time vs cmd time during stress test



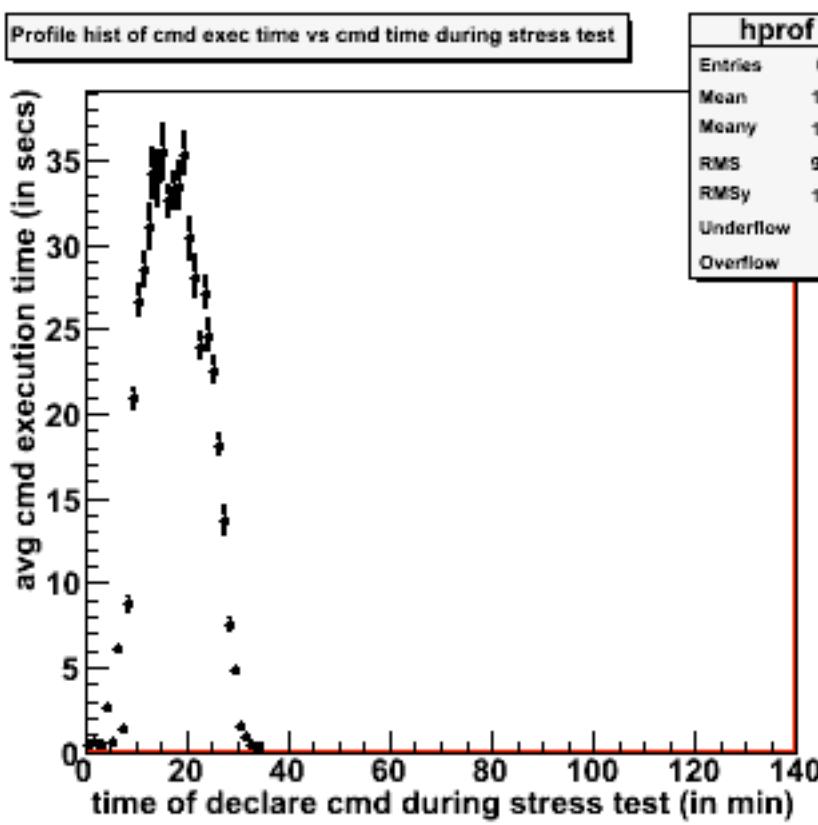
## Test Results for: Three simultaneous jobs w/ 50 sections each:



number of declare commands per minute

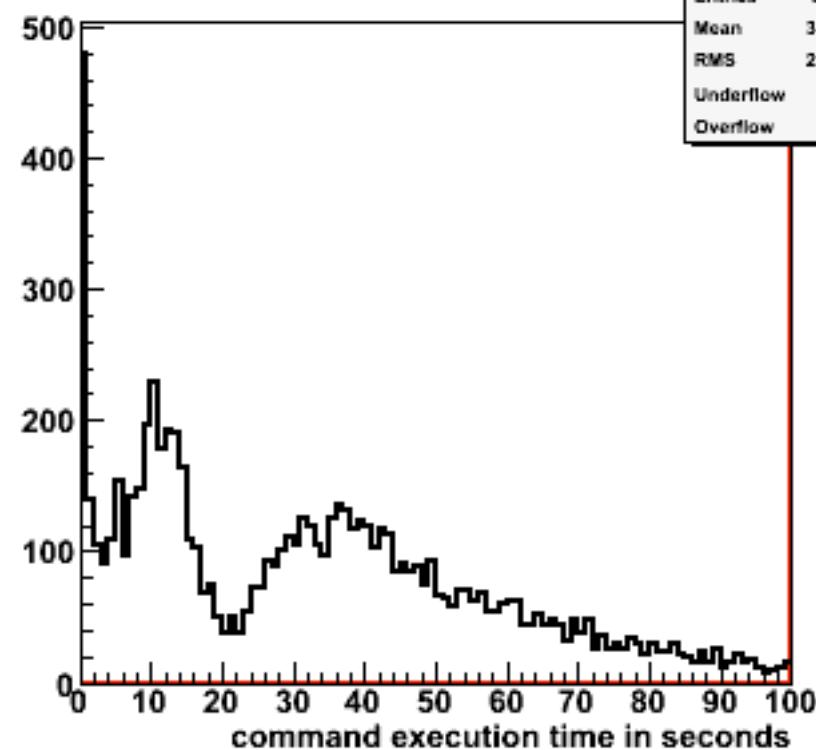


Profile hist of cmd exec time vs cmd time during stress test

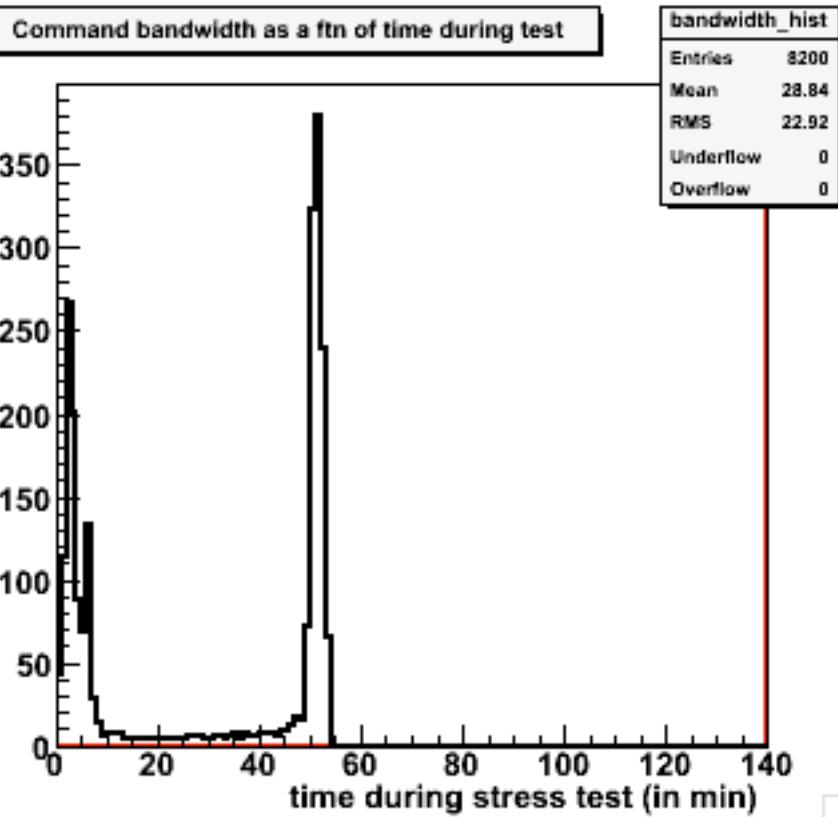


## Test Results for: Four simultaneous jobs w/ 50 sections each:

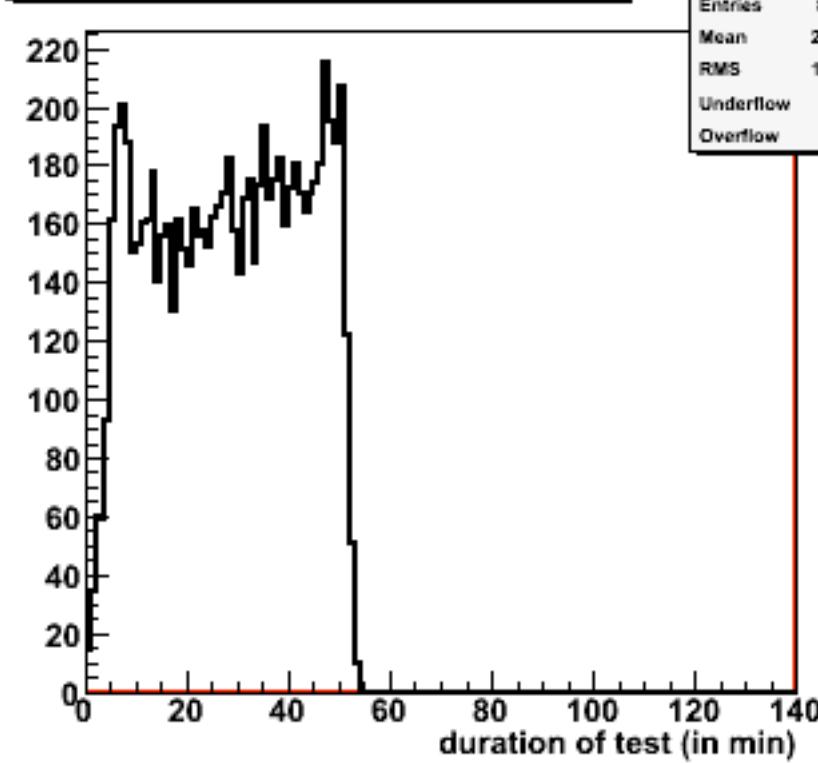
declare metadata command execution time



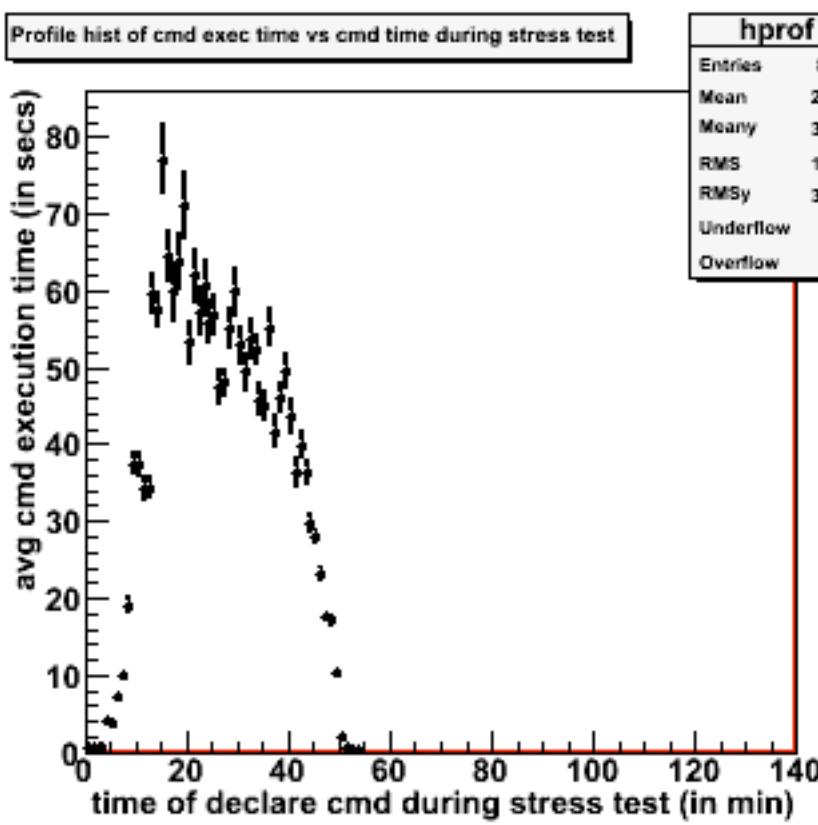
Command bandwidth as a fn of time during test



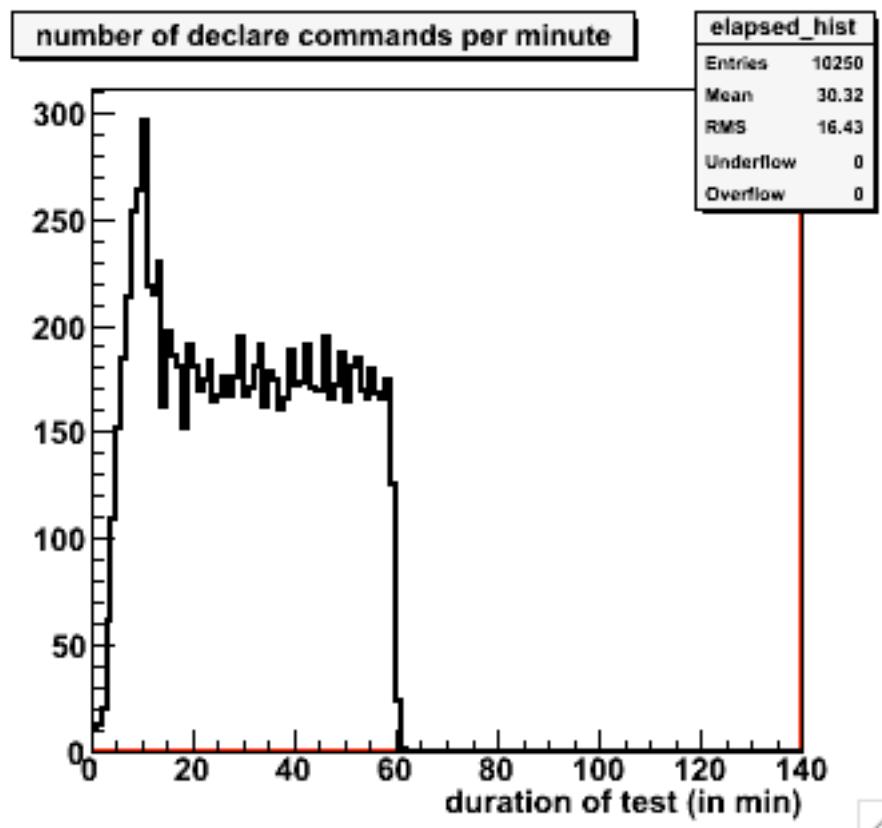
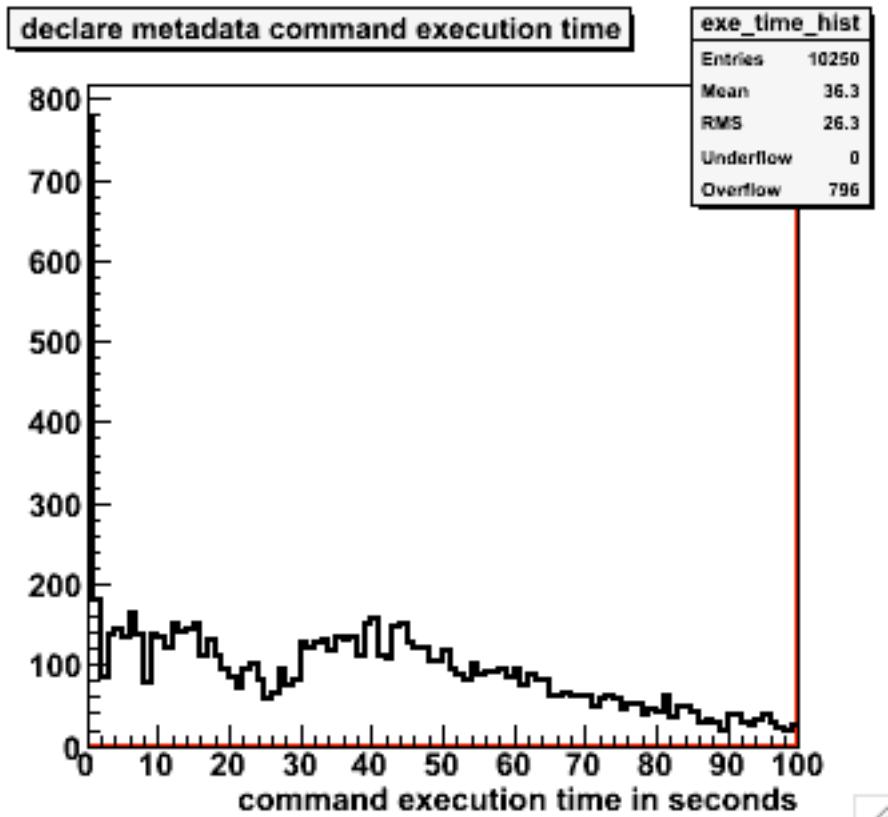
number of declare commands per minute



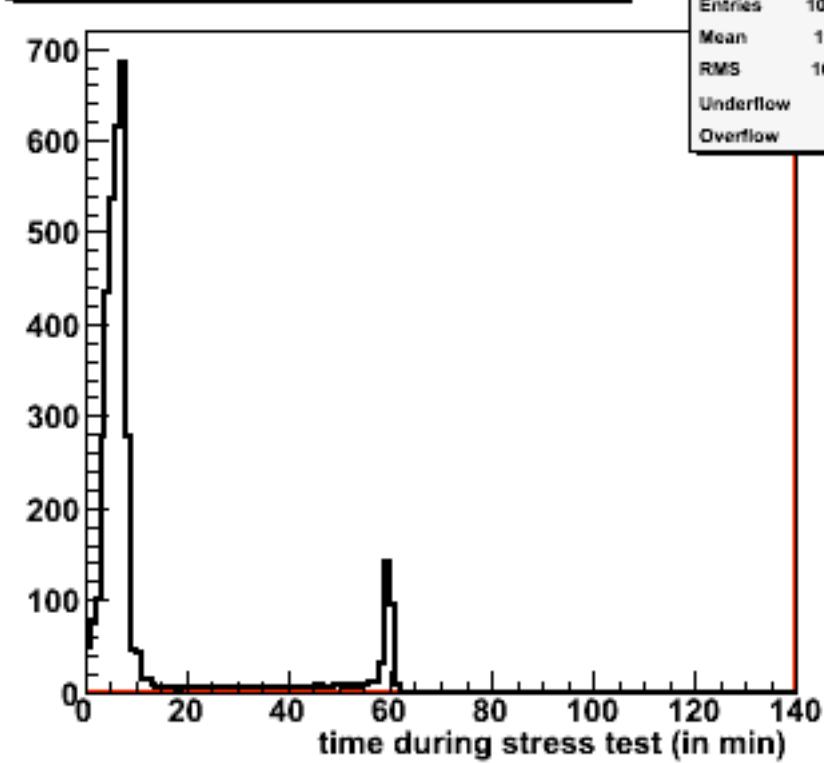
Profile hist of cmd exec time vs cmd time during stress test



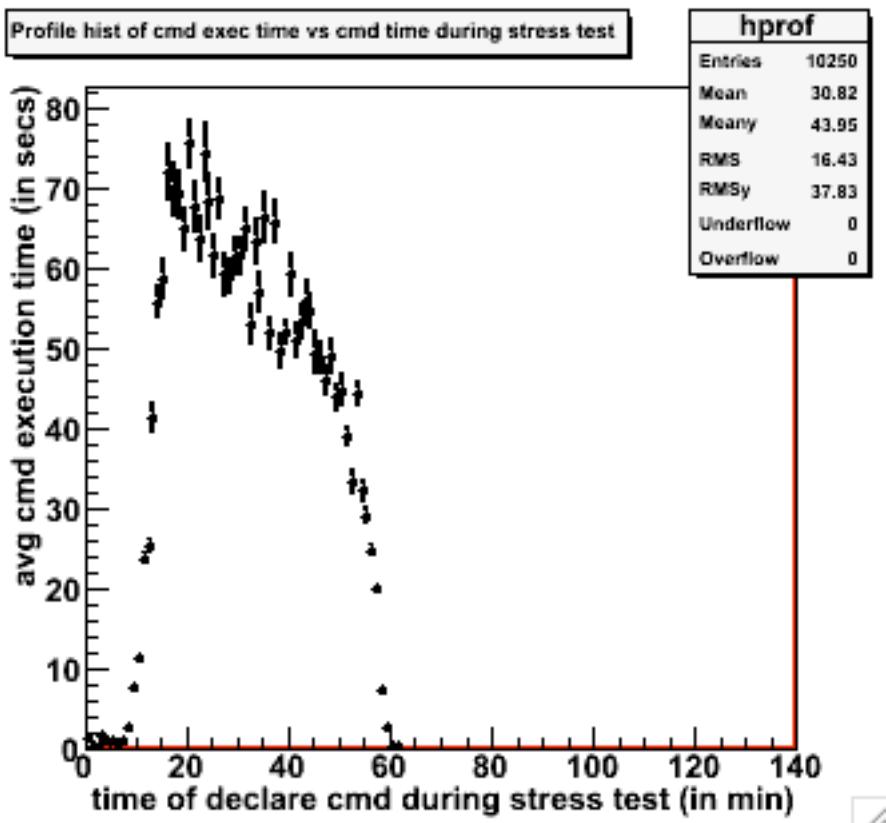
## Test Results for: Five simultaneous jobs w/ 50 sections each:



Command bandwidth as a fn of time during test

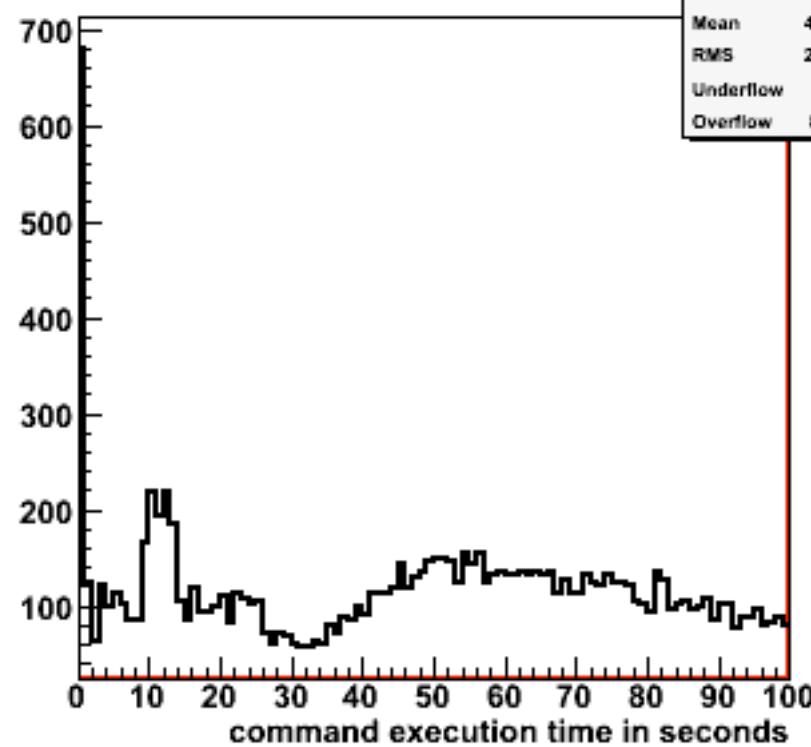


Profile hist of cmd exec time vs cmd time during stress test

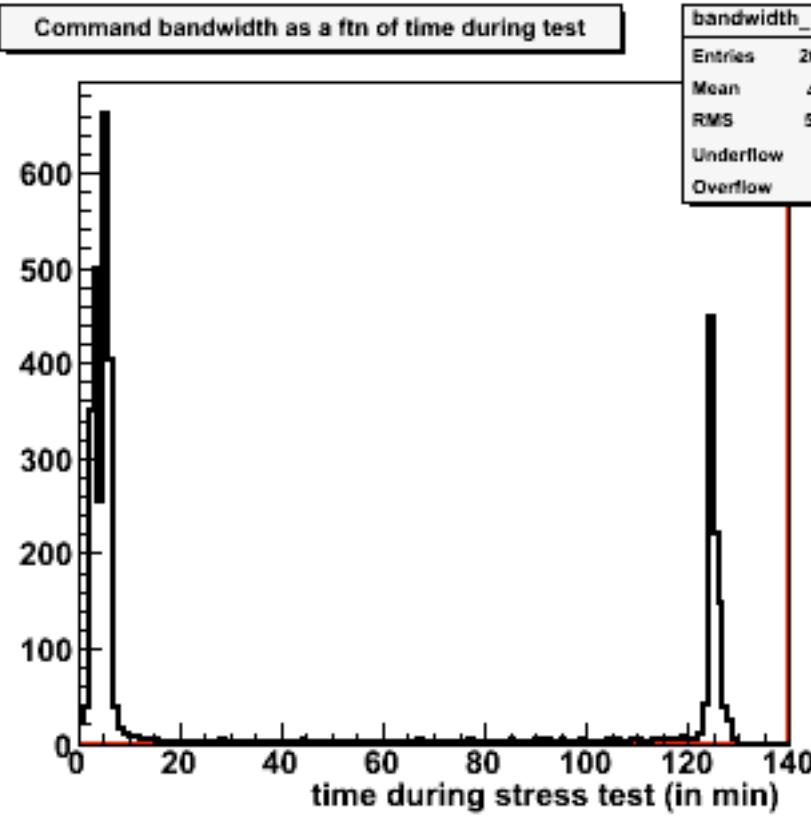


## Test Results for: Ten simultaneous jobs w/ 50 sections each:

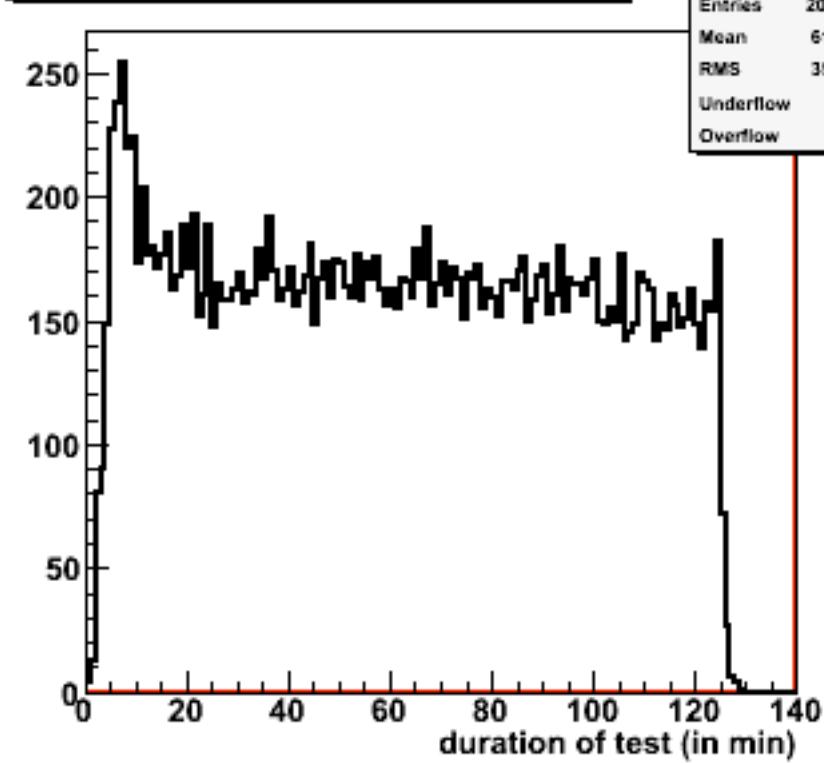
declare metadata command execution time



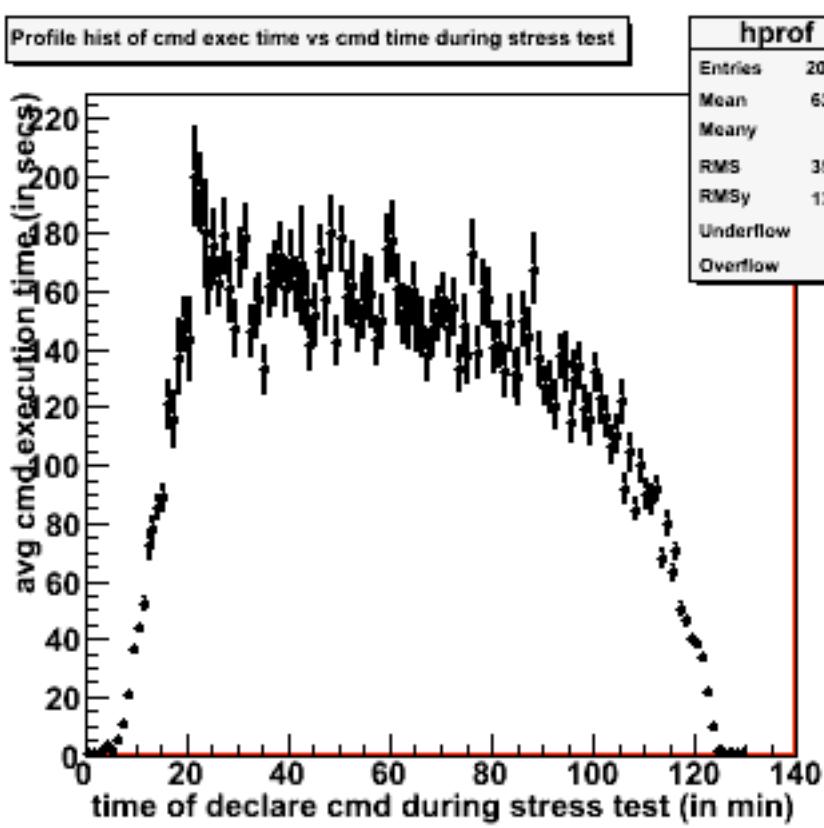
Command bandwidth as a fn of time during test



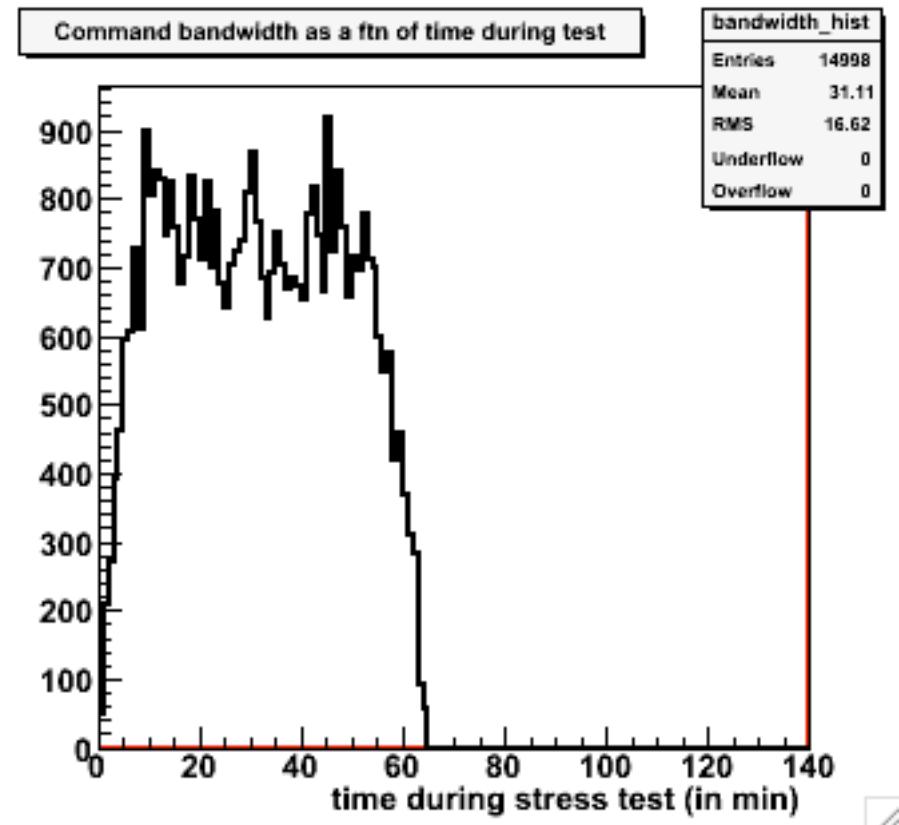
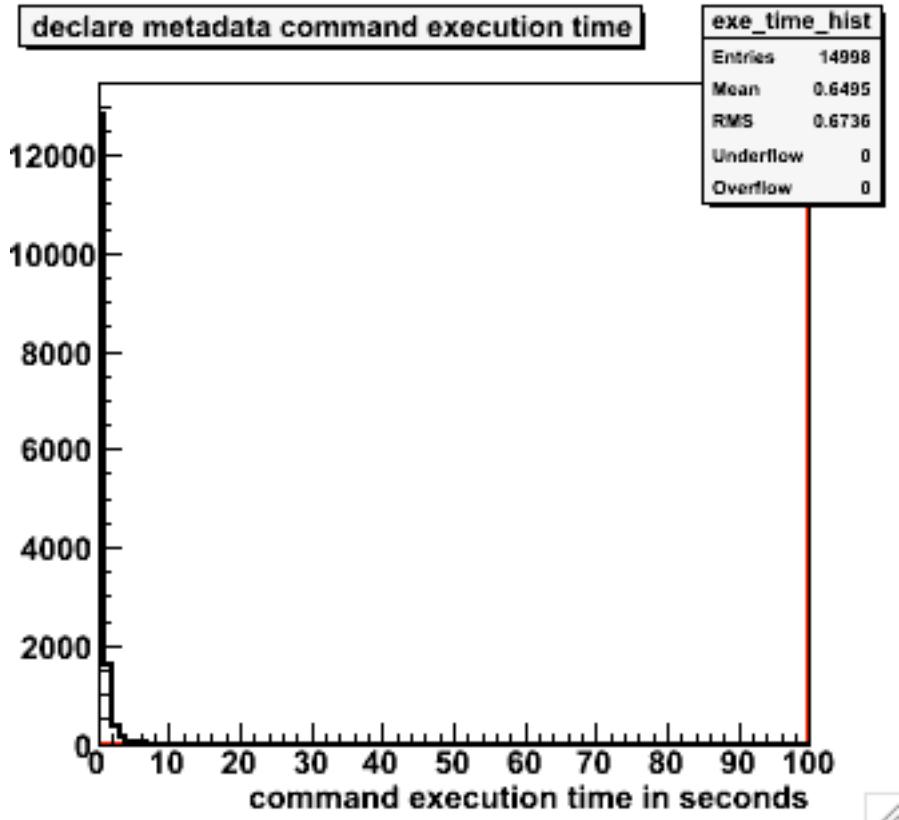
number of declare commands per minute



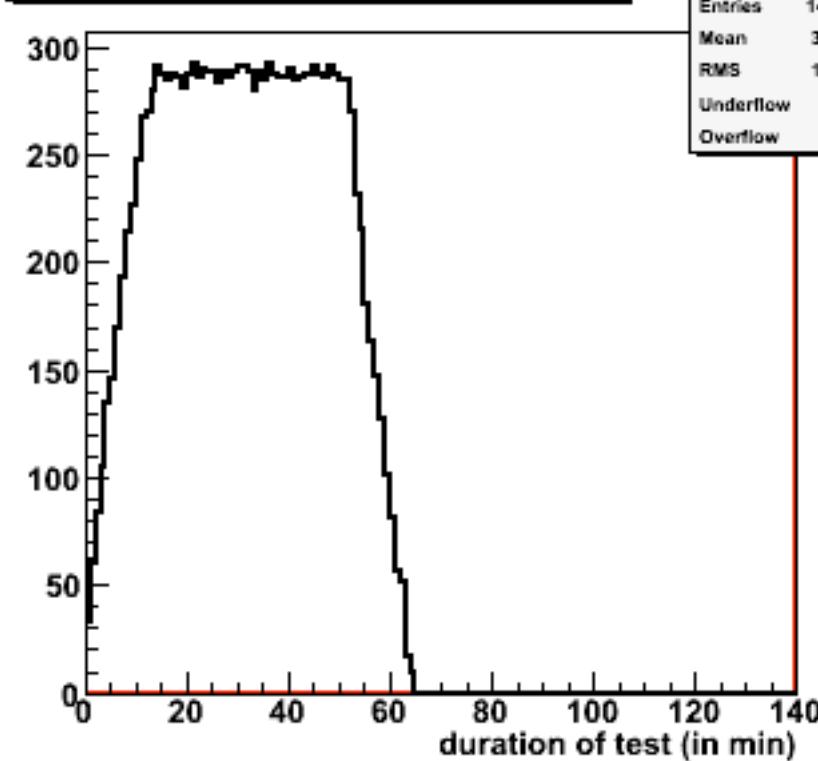
Profile hist of cmd exec time vs cmd time during stress test



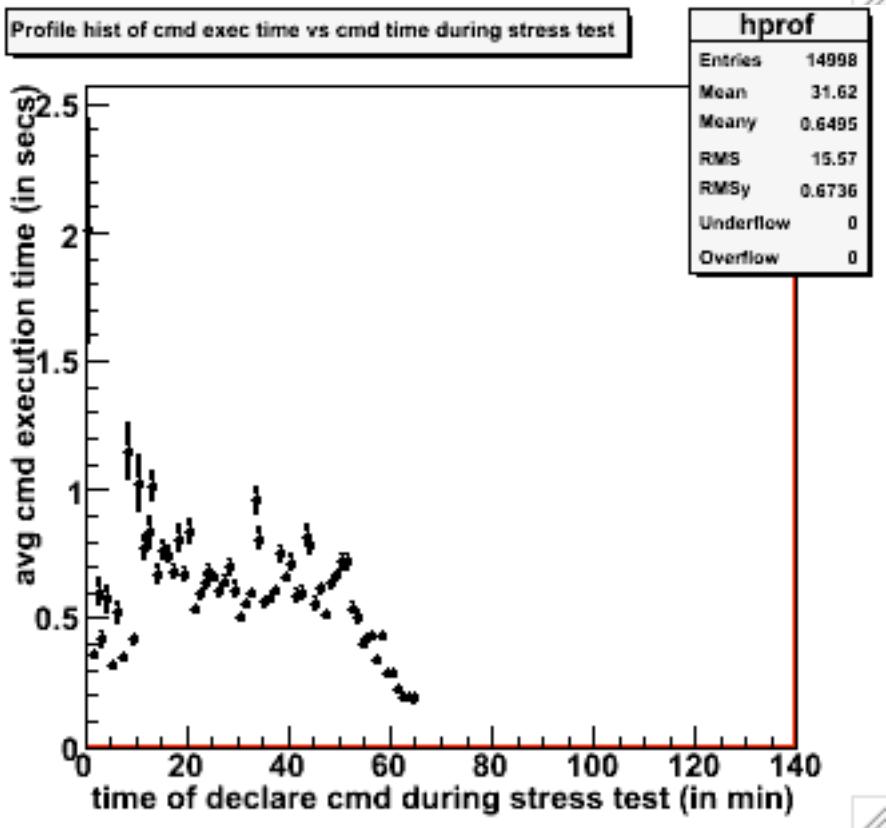
## Test Results for: 1 job w/ 75 sections;



number of declare commands per minute



Profile hist of cmd exec time vs cmd time during stress test



## **Conclusions:**

As the number of simultaneous metadata declaration increases, the amount of time to execute the declare metadata increase. Once the maximum is reached it can take a while to process all of the requests. The command bandwidth plots show that the number of completed declarations per minute drops and the requests appear to queue. The final test with 75 sections running at once (each section with 200 file metadata declarations with a sleep of 15 seconds between each declaration shows that the system can operate with reasonable performance.

## Appendix A – list of python script to test SAM metadata declaration.

```
#!/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

#
# 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=1000

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)

#
# 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'

=====
=====
```

```

# and here is the loop over the files
#
=====
=====

file_number=0
while file_number <= file_limit:
#    create the dummy file name
    timestamp = strftime("%Y%m%d%H%M%S", gmtime())
    filename = 'test_' + os.environ['CAF_JID'] + '_' +
os.environ['CAF_SECTION'] + '_' + timestamp
    metadata['fileName'] = filename

    file_number = file_number + 1
    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)

#    now sleep for 15 seconds

    time.sleep(15)

#
#    get the total elapsed time etc.
#

job_stop=time.time()
job_stoptime=time.time()

job_stoptime_string = '%s' %strftime("%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)

```