National Centre for Radio Astrophysics

Internal Technical Report

GMRT/TGC/001-Jul 2024

Reducing the TGC System dead-time between consecutive Observing data-scans

Author : Jitendra Kodilkar, Deepak Bhong, Ishwara Chandra

Email : *jitendra@ncra.tifr.res.in*, *deepak@ncra.tifr.res.in*, *ishwar@ncra.tifr.res.in*

Objective: Based on the NCRA internal users' feedback, the necessity of reducing the TGC system dead-time between consecutive GMRT LTA and Beam-former data-scans acquired using the GMRT backend was expressed. The varying dead-time of around ~3 to 4 minutes between two consecutive data-scans was putting constraints on covering the expected number of target-sources for data-acqusition in general or pulsar survey kind of observations, and was also restricting to cover total number of points specified in the grid-pointing and holography kind of experiments.

The extensive modifications in the TGC code was done to implement the broad-cast DAS (Data Acquisition System) commands, also implement the single command for starting the astronomical data acqusition ('fstartscan' or single DAS-commands like 'startscan'/'stopscan' for multiple subarrays). This drastically reduced the command turn-around time by 50% and effectively helped in reducing the dead time between consecutive data-scans up to ~ 1 to 1.5 minutes for normal observations, and 0.3 to 0.7 minutes in case of grid-pointing and holography kind of observing sessions. Thus, it is effectively now possible to conduct the survey observations, holography and grid-pointing kind of experiments using the TGC system with expected number of celestial sources within a allocated time of observation.

Revision	Date	Modification/ Change
Ver. 0.9	2024/03/21	Initial Draft Version
Ver. 1.0	2024/07/30	Report accepted in the TGC meeting

Reducing the TGC System dead-time between consecutive Observing data-scans

Jitendra Kodilkar, Deepak Bhong, Ishwara Chandra

$ Ver 1.0, July 2024 $

1 Introduction :

Based on the NCRA internal users' feedback (*reference emails Nov 2020, Mar 2021*[1]), necessity of reducing the TGC system dead-time between consecutive GMRT LTA[2] and Beam-former data-scans acquired using the GMRT backend was expressed. The varying dead-time around ~3 to 5 minutes between two consecutive data-scans was putting constraints on covering the expected number of target-sources for data-acqusition in general survey/pulsar search kind of observations, and was also restricting to cover of total number of points specified in the grid-pointing and holography kind of experiments.

In the feedback Control-systems, the '***Dead-time***' represents the amount of time between the change in controller-ouput and the initial reponse of associated process variables (PV) is visible. And the '***lag-time***' is defined as the time elapsed after Process Variables starts to change until it reaches around 0.63 (1 - 1/6) of total change of the PV value. But in the software-programming paradigm, ***'dead-time'*** refers to periods when a program is idle or not actively performing useful work due to various reasons such as waiting for the I/O operation, synchronisation of different tasks and event handling. Hence minimizing dead-time in general is critical for optimizing the performance in many types of applications.

1.1 Various delays in the TGC system : Processing-time/lag-time, delay-time, and dead-time

Time span required to cover the observation of desired number of target sources was more in the upgraded *Tango based GMRT Control* (**TGC**) system as compared to the legacy ***ONLINE*** control system. The command/observation execution time was lesser in ***ONLINE*** system because the *Master-slave* architecture of the ONLINE system is tighly coupled to the GMRT hardware systems. The data-communication packet exchange between the GMRT hardware I/O system and the Central M&C system for command execution & monitoring GMRT systems are hardcoded in the source-code it self. Whereas, the TGC system is realised using a hierarchical compoisiton of generic (or identical) control nodes which identify their role based on the configuration specified externally. In a hierarchy the Ceneral M&C (CMC) is at the top, the antenna Local M&C (LMC) system is in the middle-layer and the I/O device servers at the bottom which control the actual GMRT hardware/subsystems. TGC system is also developed as an exploratory prototype with aim to explore the architectue and TANGO **[1]** technology which is planned to be used in SKA Telescope control & monitoring system.

The loosely coupled TGC architecture which balances trade-off between developing a generic monitor & control system which can be implemented rapidly for any new-subsystems/antenna to be added in the radio telescope, and intrinsic processing-time, lag-time delay. Therefore, the ***intrinsic processing-time***

1 Emails received from Prof. Ishwara Chandra based on the academic feedback meetings, and the TGC Bi-weekly meetings.

2 **LTA** – *Long Term Acqusition* is a standard format used to acquire the interferometry science-data on target-sources at the GMRT.

delay and ***lag-time*** is inevitable in the TGC system because of various reasons, some of them are described as follows :

(i) The every identical control-node (like pipe-filter architecture) in the hierarchy realised its role as a *Central/Local M&C system* (CMC/LMC) or *IO control-device* by reading the external configuration database and files for the commands, monitoring, and warning/alarm parameters. The control-nodes also performs additional functions such as user-authentication and authorisation (based on user-role as astronomer/engineer/observer), command-validation, reading default values for the command arguments, self network IP/socket addresses, antenna and configured device names, and parent/child nodes network-addresses from the external database. Thus, the intrinsic delay is introduced for reading information from the external devices such as file-system, network-sockets, and database.

(ii) The TGC control nodes uses the Linux OS, therefore number of open-source s/w packges available in the TANGO framework (C++/Java/Python) and the supportive network, databases s/w packaes are used in the TGC system for the network-calling, data-parsing, and thread-synchronisation. The ***lag-time*** delay can occur in used open-source multi-threaded or multi-process s/w components to achive the synchroization by using the primitive locks or semaphores to be released by other threads and processes.

(iii) TGC system provide additional GUI and Scripting environment facility at the CMC and the LMC level for independent control. The command-flow uses the Tango event-driven system for broad-casting the command to the antenna and backend-systems in the CEB. Therefore, the event driven communication system may contribute the additional processing-delay for waiting events to be triggered from the python scripting/GUI to execute the commands in the command-flow chain from CMC-LMC to I/O devices.

Also, the CMC/LMC control-nodes are developed in the Java language, the I/O Tango devices are developed in C++, and Scripting-Manager is developed in the Python to achieve higher-level modifiable algorithms for the command implementation. Thus, *command-response* conversion between child-parent nodes, propogating the system monitoring/alarm parameters up to the CMC involves processing-delay in the TGC which is unavoidable at the movement.

1.2 Dead-scan Time :

Here, we consider the pure dead-time from the time when the target-source is specified in the observing command-file for a observation up to the time when the astronomical science data-acqusition is triggered for the given target-source. Therefore, overall dead-time in observing the target-source includes ***delay-time*** for positioning and ensuring that all the working antennas grouped under a given subarray are tracking the target-source. The dead-time also includes the ***process-time*** for multiple backend commands to get successfully executed for triggerring the data scan such as adding the target-source to observing project source-catalog in the Local M&C system (LMC) catalog-database, setting the phase-center source for the fringe-stopping, and then sending start data-acqusition (***start-scan***) command to the GSB/GWB backend(s) along with the required arguments (target-source mean RA-DEC, observing-frequency and side-band information parameters).

1.3 TAT and dead (Slew) scan-time Optimization :

For the changed algorithms and new s/w modules developed in the TGC system, two kind of methods are used to check it's effect on dead-time optimization. One is to note the ***Turn-Around-Time (TAT)*** using a python-module[3] which measure total time required from the time when command is issued from the TGC CMC (*Central Monitoring & Contro*l) node to the LMC node, and response of that command is received at the CMC. Another method used in validation testing is to measure the ***dead scan-time*** which is the time between previous data-scan stopped and the time when next data-scan started. The dead (slew) scan-time is measured using the ***lta_time.pl***[4] perl script.
Test observations are conducted on the close celestial sources or the same-source with 30' shift in the declination. This avoided the delay-time in antenna positioning between the target-sources that are distant apart due to the large celestial coordinates differences. For the grid-pointing and holography experiments, the target points specified by the grid-pointing/holography model were used. However, this report doesn't address the science results like expected sensitivity, measured primary beam etc., but timely feedback on the science-data validation were received from the concern users. Hence scope of the testing is constrainted only to the results from the TGC validation tests like *TAT* and *dead scan*-times.

The new developed modules or modifications done in the TGC system effectively reduced the variable dead-time between consecutive data-scans from **3 - 5 minutes** to **1.3 - 1.6 minutes** (*Mar-June, 2021*).

The dead-time is optimized by omiting faulty antennas from the '***gotosrc()'*** waiting loop which ensures that every antenna in the subarray is tracking correct source. Previous algorithm was waiting for those antennas which were having problems such as the antenna/servo is in manual mode, servo system AZ/EL brakes are not releasing, and Servo computer malfunctioning etc. The antenna tracking algorithm is also optimized by internally sending a servo-system '***abort***' command before issuing the new '*track*' command to the servo-system. The use of '**abort**' command helped in removing the stale-source coordinates of the previously tracking source in the servo-system and saved time up to ~30 seconds. Thus, the latest ***lag-time*** for tracking subarray of thirty antennas is around ~20 to 30 seconds.

The further scope of this report is more focussed on summarising the changes or new development done in the command execution algorithms of the TGC system to reduce the Turn-Around-Time and thereby reducing the dead-scan time between consecutive data-scans taken ***using the GMRT backends (GSB/GWB). The total ~18 or more than that tests were conducted, results were shared timely over the email and summary was discussed in the group-discussion to get further inputs from the academic and senior members.***

*T*o minimize the dead-time from ***~1.6 minutes to ~0.4 to 0.7 minutes*** in case of the holography & grid-pointing experiments respectively, and up to ***~1 minutes*** in case of the multi-subarrays observation using both the correlator GSB and GWB, we applied various techniques such as : **(i)** ***Asynchronous execution*** to efficiently schedule tasks of antenna-tracking and backend-commands. **(ii)** Instead of serially sending the TGC commands to GSB and GWB, ***broadcast commands at a time***, and aggregate the response.

3 Module for TAT for every required command is incorporated in the scripting-manager developed by shri. Deepak Bhong.

4 ***lta_time.pl*** perl script developed by shri. Santaji Katore. Furhter modified to skip the bad-scans by Jitendra Kodilkar

(iii) Making a provision of *time* (*hr:min:sec*) argument for the start data-acqusition command so that the TGC command can be ***executed in advance but the data-acqusition trigger will take place in the future time*** specified by the the *time* argument. **(iv)** Optimizing the multiple data-acqusiton commands into a single '***fstartscanproj***' (fast scan-project) for adding the source , setting the phase-center source at the LMC, and then finally starting of the data-acqusition.

Section-2 mainly addresses the new development/modifications in the TGC ***, section-3*** gives details of the test experiments and results validation, ***section-4*** describes details of the functional problems occurred in the TGC and how it is resolved, and the last ***section-5*** gives summary and conclusions.

2 New developments and Modifications in the TGC-system

The back-end Configuration and Data Acqusition System (DAS) commands in the TGC system for the GSB and GWB were use to be issued serially due to inherent different arguments required for individual DAS commands. The serial execution of DAS commands introduced the unwanted dead scan-time in survey, grid-pointing, and holography kind of observations. The TGC codes at the CMC, LMC (Java, Python API), Tango I/O control-nodes servers (C++), and Client-programs running on the GSB/GWB machines (C++) are extensively modified to incorporate the new algorithms developed in order to reduce the dead-time between the two consequent scans in the LTA data and the TAT (Turn-Around-Time) for the backend commands in the TGC.

***Annexure-I** tables gives the details of new developments and code modifications. The subsections given below briefly mention the functional changes done in the TGC code.*

2.1 Asynchronous Task-scheduling :

The addition of source-catalog, digital-backend (GSB/GWB) configuration for the scheduled observation, observing project creation, and tuning of the receiver-chain systems of the antenna (Front-end Common-Box, GAB, Optical-Fiber etc) in general is done at the begining of observing session. Whereas the execution block for observing source called 'scan' is iterated with a pre-defined number of commands which mainly track the antennas on given target-source, and send commands to the backend systems for startinng/stopping acquisition of the science data. Hence, the dead-scan time optimization is done by asynchronously scheduling and executing commands.

The asynchronous task-scheduling is acheived in two ways :

(i) The observing batch-script in Python environment do not wait for the response of issued command beyond the predefined command timeout period is over. Therefore, the ***track_array()*** command for a given sub-array in the observing batch-script is specified with ***time-out*** argument having a limited pre-defined period of ***~ 5 to 10 seconds***. In this case, the observing script do-not wait for the response of given command but antennas continue its processing like antenna positioning & tracking. The CMC receives only final aggregated response from all antennas. While the antennas are positioning on target-source, the preparatory commands for starting the data-scan like adding and setting of the phase-center for a given project are issued and executed at the backend/correlator LMC system. And after checking that working antennas are tracking correct-source, the TGC system issue the start-scan command to the GSB/GWB backend immediately.

(ii) In case of the **VLBI** (Very Long Baseline Interferometry) observing sessions, it is critical to synchronise the time of starting of the data-acqusition scan at the correct time in order to align the data taken with the other telescopes. For this purpose, the ***start_proj()*** or ***fstartscan()*** *c*ommands accept the "***time_str***" argument in *HH:MM:SS* format so that the data acquisition command gets executed in advance but the acqusition trigger of the science data take place at the given ***'time_str'*** argument. By executing the start_proj/fstartscan command with the future expected time to trigger the data acqusition, rest of the time can be reschedule to position and track the array on a given target-source. Thus, this kind of asynchronous execution also help in reducing the dead-time between two consecutive data-scans.

The command for starting/triggering the data-scan commands with the 'time_str' arguments has multiple options such as post-pone or prepone the command-execution, and abort the data acqusition command as well. The standard operating procedure for starting the data scan is as given below :

- **start_proj ('BOTH/GSB/GWB', <Sub-array No.>, time_str="HH:MM:SS")**
  // "time_str" argument is optional, if not given, command is executed like any other normal
  // command which start the data-acqusition instantly when command received.

  *// The 'time_str' argument is also in a similar manner for the **'fstartscan'** command described in the **section 2.2***

- If the **'time_str'** option is given, then the 'start_proj' command gets buffered in 'deviceclient' program running on the digital backend machines, it's execution behavior is as follows :

  (i) if 'time_str' > *current_time :* Backend will send success response to the CMC of TGC, and trigger the command when *current_time* <= *'time_str'*.

  (ii) if 'time_str' <= current_time : start data-acqusition command executes immediately, as if no *time_str* argument is given.

- If the 'start_proj' Command is repeated with the new "time_str_next" again :

  (a) if time_str_next > time_str_stored , then it will overwrite the time, and execute whenever ***time_str_next*** <= current_time.

  (b) if "time_str" argument not given OR **'time_str_next' < time_str_stored**, command will be executed when the current_time >= ***time_str_next***.

- **Upon issuing correlator 'halt' command or command with 'abort' option i.e.**

```
execute_command('GSB/GWB', corrctl1, 'abort');  // All the buffered/planned 'start_proj'
                                                // command will be canceled.
```

2.2 Single fast data-scan command - "fstartscanproj" :

To trigger the data acqusition-scan on target-source, TGC system comprises serial execution of subsequent three commands such as **(i)** Add phase-center source in the backend LMC catalog, **(ii)** Configure the fringe-stop parameters by selecting the phase-center source information from the already added backend LMC-catalog along with the frequency parameters. **(iii)** And finally issue a 'startscan' command.
The required total turn-around time (TAT) for 'add-project source', 'set phase center source', and the 'startscan' command execution is **~ 50 seconds**. Whereas, if the phase-center source is already present in the LMC-catalog, then the configuring the phase-center by the 'set phase center souce' command and issuing 'startscan' command i.e. only two commands execution TAT takes only **~36 seconds.**

Instead of executing two to three commands serially for starting the data-acqusition per subarray, a single command is created in the TGC system named '***fstartscanproj***'. This single command perform addition and setting of the phase-center, and issuing of 'startscan' command in one go which takes only **~ 18 seconds**. Thus, using the 'fstartscan' command , turn-around-time for starting the data-acqusition is reduced by **50 %.** The **Figure-1** shows measured Turn-around-time (TAT) with two to three serial DAS-command execution for the data taken in Apr-May 2023 is comparied with the Turn-around-Time (TAT) of '*fstartscan*' command for the data taken in Sep-Aug 2023. The consistent data taken over long duration shows that *~40 to 50 seconds* of TAT required for the default serial das-commands has been reduced to ~20 seconds only for the 'fstartscan' command.

- The syntax for '***fstartscanproj***' is as follows :

fstartscanproj < **subarray id** > , **op_short=0** (*0=default-source, 1=setphase_source, 2=add+setphase*) <**source_name**> **op_short**=0 *(1=set-TPA frequency)* **<RF1> <RF2> <FIRST_LO1> <FIRST_LO2> <BB_LO1> <BB_LO2> <bandmask> <rest_freq1> <rest_freq2> <ch_width> <qual> <integ>** <time_str> (*for advance command in hh:mm:ss*) <draddec_reftime> *(in hrs for planetary object tracking)*

(i) op_short = 2 *// Automatically select to 2 if phase-center source is not added in the // backend 'LMC-catalog'.*
= 1 *// Automatically select to 1 if phase-center need to be configured*
= 0 *// If the same phase-center is repeated for the starting the data-acqusiton scan // i.e. no need to configure.*

(ii) op_short *= 0 // No need to configure the fringe-stop RF frequency to the correlator*
= 1 // Set the fringe-stop RF frequency and other parameters

(iii) time_str = 'HH:MM:SS' *// Optional - argument to trigger the data-acqusition scan in expected time.*

(iv) dradec_reftime = <hrs> *// Optional - Reference time for ΔRa, ΔDec for planetory objects.*

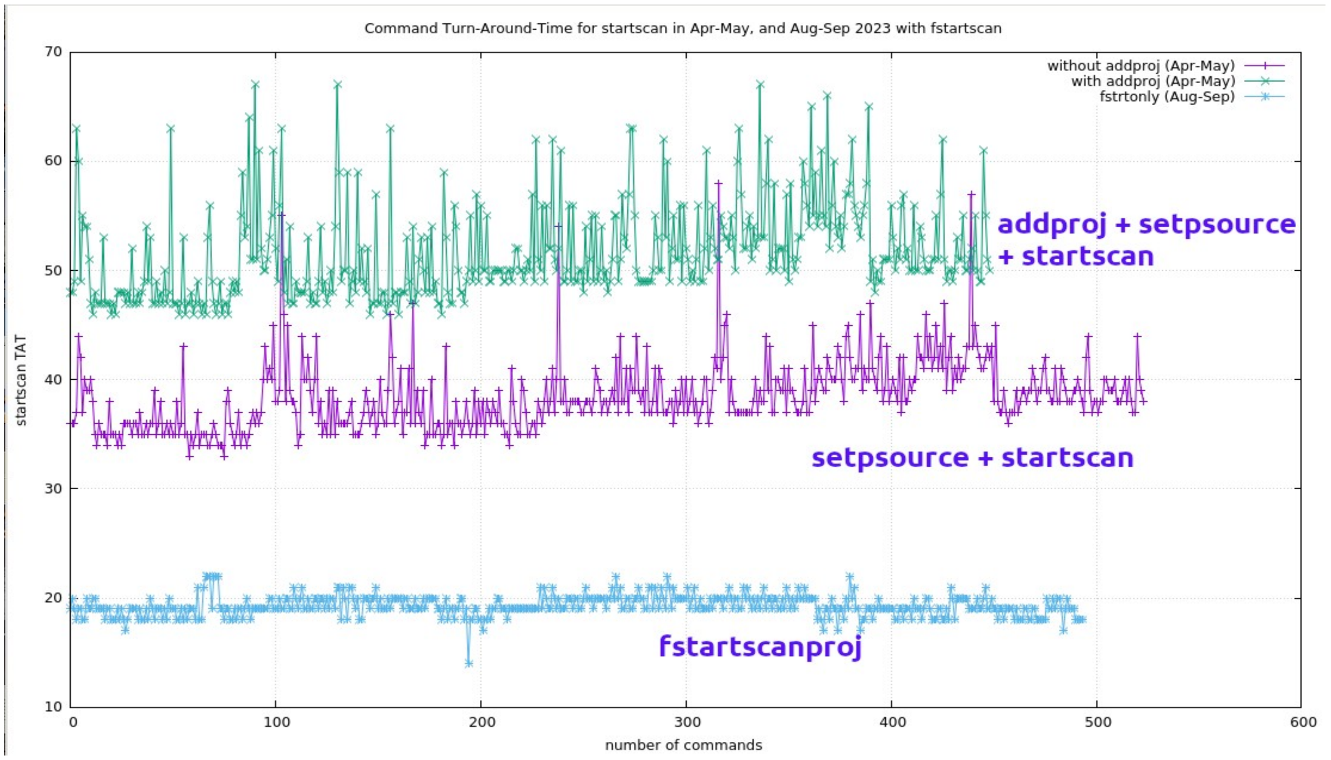


***Figure** 1: Command Turn-Around-Time (TAT) in Apr-May (Serial commands) and Sep-Aug ('fstartscan') 2023*

2.3 Broad-cast DAS command implementation :

The legacy ONLINE system issue the data-acquisition system (DAS) commands serially to the GSB and GWB backends, and even for each subarray separate commands are issued in case of multi-subarray observation. This is because DAS command from the ONLINE were issued to the correlator using the client-server synchronous communication. Therefore, initial development of the TGC also followed the same logic mainly because two backends (GSB and GWB) takes different types of fringe-stop arguments. In case of multi-subarray observation (two subarrays), the turn-around-time (TAT) in the TGC system was around ***~2 to 3 minutes.***

To reduce the command TAT and dead-time between two scans, the TGC software modified so that the correlator commands can be broad-casted simultaneously to both the correlator, and also instead of issuing separate start/stop-acqusition commands to each subarray, only one command is implemented in the TGC to start/stop the data-acqusition for the more than two subarrays. ***Thus, the parallelism in command execution reduced the Turn-around time from 135 - 174 seconds to ~70 seconds.*** **Figure-2** shows the average statistics of turn-around time measured in the TGC system for data-acqusition commands issued to the backend system.

To implement the broad-cast command method, the CMC, LMC , IO device-server code and the deviceClients programs running on the backend-machine are modified. **Table-1 A** and **Table-2 B** describe the functional changes done in the TGC backend related commands which are as follows :

TABLE-1 A: The data-acqusition commands to both the backends are broad-casted in the TGC using a Single command with multiple arguments for each subarray :

#	Command	TGC Implementation
1	**getpsource**	*// Broad-cast the set phase-center source command to both the GSB and* *// GWB backend using a 'getpsource' which can take arguments for two or three* *// subarrays simultaneously i.e. subarray numbers separated by '#' character* *// i.e. '1#2', and phase-center source-name '3C48' and '3C286' respectively.* **execute_command('GSB,GWB', 'getpsource', "1#2, 3C48#3C286")**
2	**setfreq**	*// Set the fringe-stop frequency parameters to the individual correlator but* *// simultaneously for two or three subarrays. Example given below set the* *// frequency parameters for two subarray (1 and 2) in one go using a single* *// command.* **execute_command('GWB','setfreq','1#2','1460#500','1460#500','-1460#-500','-1460#-500','0.0#0.0','0.0#0.0','12#12','0.0#0.0','0.0#0.0','1#1','0#0','1#1')**
3	**startscanproj**	*// Broad-cast command to start data-acqusition for both the GSB and GWB* *// correlator, and two sub-array (subarray 1 and 2) simultaneously.* **execute_command('GSB,GWB', 'startscanproj', '1#2' , '00h00m00s#00h00m00s','0.0#0.0')**
4	**stopproj**	*// Broad-cast command to start data-acqusition for both the GSB and GWB* *// correlator, and two sub-array (subarray 1 and 2) simultaneously.* **execute_command('GWB,GWB', 'corrctl1','stopproj','1#2')**

TABLE-1 B : Scripting APIs to be used in the observing file at user level are developed by shri. Deepak Bhong for the above commands, which are as follows

#	Scripting Command Example	Description
1	**set_source_corr([1,2],'3C286,3C123','GSB,GWB')**	Add and configure the phase-center source for Subarray-1 and 2 of both GSB and LMC backend.
2	**strtndas('GSB,GWB','1,2')**	start the data-scan for multiple subarrays and multiple backends in one go.
3	**stpndas('GWB,GWB', '1,2')**	stop the data-scan for multiple subarrays and multiple backends in one go.
4	**fstart_proj('GSB,GWB',<subarray No>,<source-name>)**	fast-start scan interface with **broad-cast facility** for single-subarray.

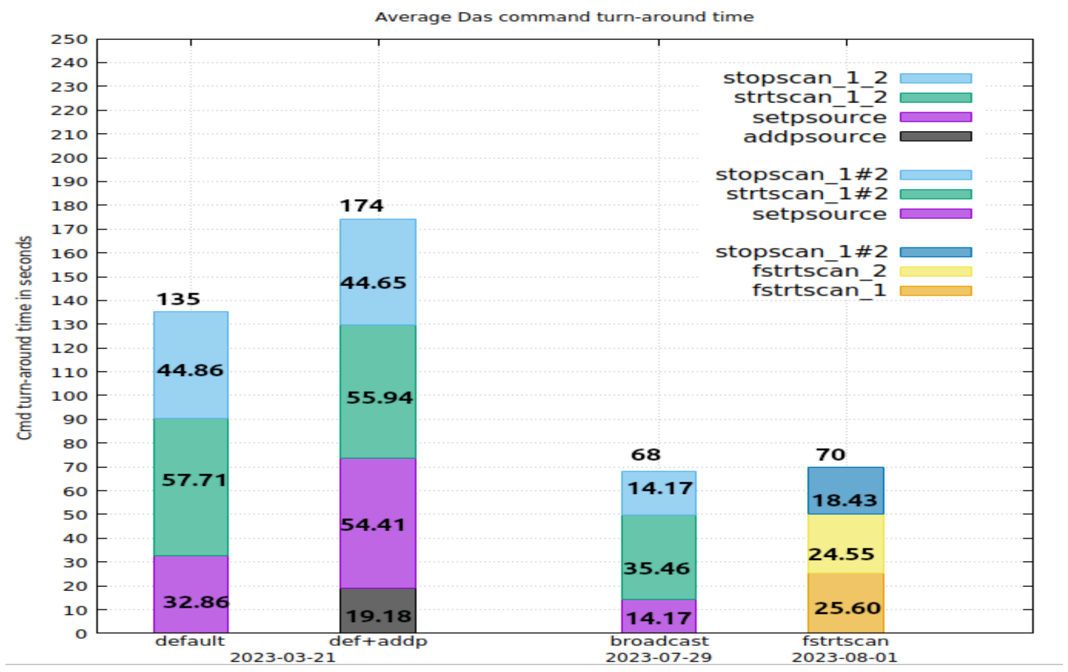


Figure 2: *Comparison of Turn-around time for data-acqusition commands in serial/default mode, Broad-cast (parallel) and fstartscan*

2.3.1 Effectively handling of the Command-failure :

The CMC , LMC level codes modified to aggregate the response for data-acqusition broad-cast command which is issued to the multiple projects running for both the backends. Effectively, using the scripting modules it became possible to check the failure responses from the project running under individual sub-array(s) and backend(s) so that the command can be retried only to the failed sub-array of particular backend.
If the aggregate response of broad-casted command is received failed, then the data-acqusition status of individual project is checked (which is already updated by the LMC system), and command is retried only for the project for which command to start/stop the data-acqusition is failed.

3. DAS Command Testing and Dead-scan optimization Results :

In the TGC system, number of tests were conducted to measure dead-scan optimization time and Command turn around time (TAT) before and after the code modifications implemented for the Data Acqusition System (DAS) commands ***viz. 'addproject', 'setpsource', 'setfreq' , 'startscan', and 'stopscan'.***

The dead-scan time between the two data-scan is measured using the '***lta_time.pl***' script which measure the time-difference between the when scan was stopped and next consequent data-scan recording started at actual in the LTA file. Whereas the command ***turn around time (TAT)*** is measured using the '***gettime()***' python module in the MNCScripting manager which measure the total execution time of the DAS command i.e. total time required when the command is issued from the Central Node to the bottom level correlator DAS program which executes that command and send the response back to the CMC node.

The subsequent sub-sections give details of various experiments conducted which show the fullfillment of the various objectives of science and engineering experiments which requires the time optimization.

3.1 Grid-pointing and Holography Experiments using the *'fstartscan'* command :

3.1.1 'fstartscan' command TAT optimization, and time-saving in the grid-pointing antenna procedure :

In this sub-section, total four experiments were conducted from ***Jan 11 to Mar 13, 2023*** to validate the dead scan-time optimization and the Turn-around-time command reduction by comparing the dead-time and TAT required for the default serial commands ('addprojsrc' + 'setpsource' + 'startscan') and the dead-time, TAT required using the new single command 'fstartscan'.

From experiments conducted shown in **Table-2** , the resultant TAT for default serial commands and the 'fstartscan' command for single backend (GSB or GWB) are noted.

TABLE - 2 : System test and grid-pointing experiments test using the single Backend

#	Date	Experiment Type	Description	Turn around time, and Dead-scan optimization time
1	**11 Jan 2023**	System Test	The TAT measuring for the GSB backend using single sub-array	**(i)** TAT = 40 seconds for startscan command ***addproject + setpsource + startscan.*** **(ii)** TAT = 29 seconds without addproject command. **(iii)** TAT = 13 seconds using the 'fstartscan'
2	**08 Mar 2023**	System Test	'***fstartscan***' command options enabled for **(i)** addproject + setproject, op_short=2 **(ii)** setproject only, opt_short = 1 **(iii)** setfrequency, op_short = 1 / 0	***TAT*** is ~13 to 15 seconds with all possible options selected for 'fstartscan' command
3	**10 Mar 2023**	System Test	Turn around time measurement with both the backend GSB & GWB, fstartscan serially.	Individual backends **(i)** (addproject ~16 sec + setphase ~ 18 sec + start-scan ~ 13 sec) **TAT takes 44 to 47 seconds.** **(ii)** 'Fstartscan' command **TAT = 12-13 seconds**
4	**13 Mar 2023**	grid-pointing	Grid-pointing with the default backend commands (addproject + setphase-source + start-scan) and using the 'fstartscan' executed.	(i) **Default command TAT = 26** seconds per scan. **'fstartscan' command TAT = 13** seconds to **18** seconds. Thus, ***50% TAT is reduced.***

#	Date	Experiment Type	Description	Turn around time, and Dead-scan optimization time
				(ii) Total <u>dead-scan time</u> for both the axis : **(a)** Default total 16.5 min **(b)** ‘fstartscan’ total ~ 12 min. **(iii)** For both the AZ and EL axis, **(a) Total pointing time is ~26 minutes** in default mode. **(b) Total pointing time is ~ 21 minutes** Using the ‘fstartscan’ command ***<u>Thus, 4 to 5 minutes are saved in pointing using both the axis.</u>***

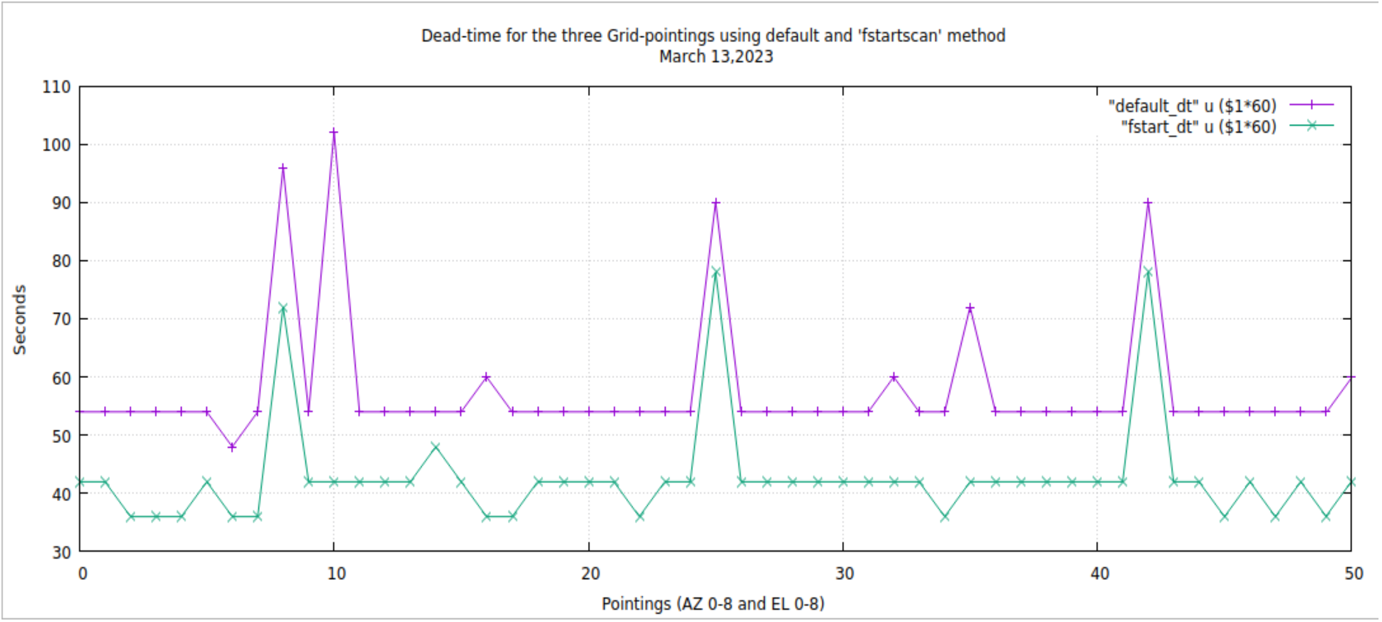


Figure 3: *Dead scan-time per scan for three grid pointing in both the axis using data taken on Mar 13, 2023*

- **<u>Validation test results</u> :**

(1) With the help of ‘fstartscan’ we could reduce the TAT by 50 % . The TAT for the default-serial commands use to take ~40 seconds (~29 seconds without ‘addproject’ command). Whereas using the ‘fstartscan’ command, the TAT is found to be ~13 to 15 seconds.
(2) The dead scan-time in the default serial mode command is **~ 54 to 60 seconds** per scan, whereas for the ‘fstartscan’ it is **36 to 40 seconds. Figure-3** show the dead scan-time per point using the three grid-scans across both the axis (AZ and EL) of antenna.

(3) The total grid-pointing time for both the AZ and EL axis using the GSB or GWB backend reduced by 4 to 5 minutes using the ‘fstartscan’ command. Thus, total time for the pointing takes ~ 26 minutes whereas the ‘fstartscan’ command takes ~ 21 minutes.

3.1.2 Grid-pointings on Multiple Hour-angle sources for the pointing Model :

To measure the Azimuth-Elevation dependent pointing model, it is required to do continuous grid-pointing experiments on multiple sources of various transit-times using different calibrator sources with multiple HA and varying declination from north to south. The long-term grid-pointing observation consists of more than 200 pointings, and hence time reduction of completing such observation was a crucial requirement.

Table-3 gives details of such experiment conducted as an example where TGC system faced missing scan problem on May 29, 2023 (Observation no. 5), after modifying to handle the scan-failure, no missing scan was noticed in May 30, 2023 data (Observation no. 6).

TABLE - 3 : Long term grid-pointing experiments with multiple HA sources for the pointing model (Experiments by S. Roy)

#	Date	Experiment Type	Description	Turn around time, and Dead-scan optimization time
5	**29 May 2023**	Grid-pointing on multiple sources for pointing model. **Total scans : 279**	The grid-pointings on multiple sources with different (HA) for the pointing model. The data is taken using the GSB backend & ‘fstartscan’ command.	**(i)** For the command ‘fstartscan’, average dead-scan time is ~0.6 to 0.8 minutes **(ii)** **Problem :** Out of total 279 pointings in grid procedure, 7 scans are missing.
6	**30 May 2023**	Grid-pointing on multiple sources for pointing model	The grid-pointings on multiple sources with different (HA) for the pointing model. The data is taken using the GSB backend & ‘fstartscan’ command.	**(i)** After fixing the missing scan problem in the PyScripting API, ***no missing scan found in total 206 pointings.*** **(ii)** ‘fstartscan’ TAT is less than one second.

- **Validation test results :**

(1) Using the ‘fstartscan’ command, the dead-time reduced up to 0.6 to 0.8 minutes. Thus over all, for large number of pointings, **the total experiment time at least is reduced by 1 to 1.25 hrs (Refer Figure-4, Plot-1)**
(2) For the pointing model experiments, out of large number of pointings (~200 to 279) five to seven scans were missing in the TGC system, this problem resolved and no missing scan found in the experiments conducted with large number of grid-pointing experiments **(Refer Figure – 4, Plot-2)**

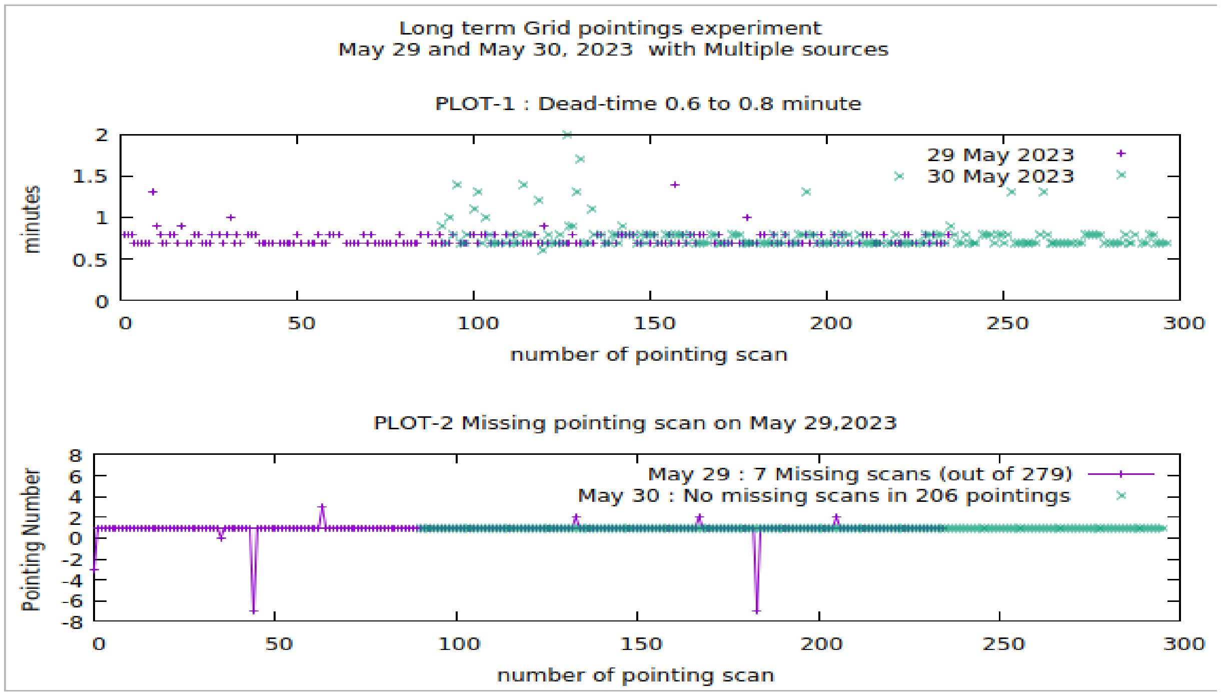


***Figure 4 :** Long -term grid pointing experiments depcting the dead-scan per pointings, and missing scan problem resolved*

3.1.3 Holography Experiment :

The Holography experiment is conducted to measure the GMRT antenna beam-pattern by taking the visibility data on calibrator at predefined grid-pointing location. This experiment use to be conducted in the legacy ONLINE which was executing the Servo-slewing command per grid-pointing and recording at least ~36 to 42 seconds of LTA data with 6 to 7 records within a one minute of boundry i.e. total one minute for the grid-pointing. The TGC system previously use to take more than one minutes of time (~ 1.5 to 2 minutes), hence the holography experiment was unable to conduct within a given allocated time for the experiments. To resolve this problem various tests are conducted to optimize the time, mainly (i) use the 'fstartscan' command, execute the track and DAS commands asynchronously. **Table-4** gives the details of these experiments with details of problem faced, and time optimized in terms of dead-scan time and TAT.

TABLE - 4 : Holographys experiment to measure the beam-pattern of the GMRT Antenna *(By Dharam V Lal/S. Katore/JPK)*

#	Date	Experiment Type	Description	Turn around time, and Dead-scan optimization time
7	**04-05 July 2023**	Holography experiment	Holography compatability test with the legacy ONLINE system using two methods : **(i)** 'fstartscan' command with advance time to trigger the scan, and then give array-track command.	**(i)** As compared to method-1 ('fstartscan' command first and then array-track), method-2 has less noisy data. **(ii)** In both the case, number of records collected per scan are compatible with the legacy online i.e. ***<u>In total 1 minutes spend per-grid</u>***

#	Date	Experiment Type	Description	Turn around time, and Dead-scan optimization time
			(ii) Array track command first, and then 'fstartscan' command without advance time.	***point, total recording-time is 36-42 seconds and dead-time scan-time of 18 to 24 seconds.*** **(iii) 4 July 2023 data problem –** Pointing name is correct but *coordinates of previous pointing is repeated.*
8	**10 July 2023**	Holography experiment	Holography test to check compatibility with the ONLINE for total 1 minute per scan-point	**(i)** TAT for 'fstartscan' 17-19 seconds + 2 seconds for 'stopscan'. **(ii) *Dead-time is 0.3 to 0.4 minute*** **(iii) *Number of records are 6 to 7 i.e. 36 to 42 seconds .*** recording. Scan execution happening per one minute.
9	**21-22 July 2023**	Holography experiment	Holography test with 2 minute per pointing.	**(i) Dead-scan time is 0.3 to 0.4 minute,** Total-Recording time is 1.6 to 1.7 minutes. **(ii) Scan execution trigger 2 minutes exactly.** **(iii) *Source-coordinate repeatation occurred on July 4, 2023 is resolved.*** - ***resolved. resolved.***
10	**25 Aug 2023**	Holography experiment	Holography test with 2 minute per pointing.	

- **Validation test results :**

(1) Using the TGC system with 'fstartscan' command, the holography experiment could conducted successfully with the 1 minute of boundry per grid-pointing along with the ~36 to 42 seconds of recording time and dead-time 0.3 to 0.4 minute. Thus, dependancy of conducting the holography experiments using the legacy ONLINE is removed. **Figure-5** shows the test results conducted on July 10, 2023 (*Observation no. 8) .*
(2) To increase the sensitivity per grid-pointing, holography experiments were conducted with the 2 minutes of execution time of boundry per point. **Figure-6** depicts the dead-time is ~0.3 to 0.4 minutes, and recording time ~1.6 to 1.7 minutes with 2 minutes of boundry for execution per grid-pointing. The data used for this plot was taken on July 21st, 22nd, and Aug 25th , 2023 (*Observation no. 9 and 10*).
(3) The Holography experiments science test results are accepted using the TGC system, and no need to use the legacy ONLINE system.

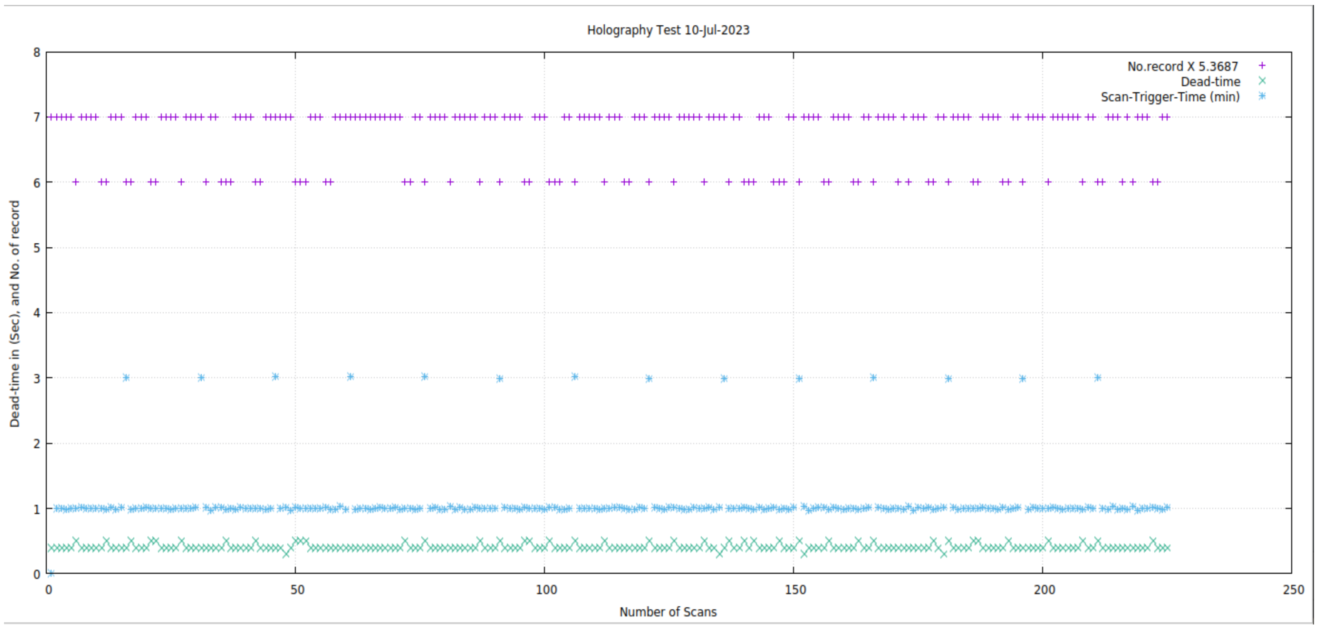


Figure 5 : Compatibility of the TGC system with the Legacy Online (Mar 10, 2023 data):   1 minute of execution per grid-pointing with dead-time 0.3 to 0.4 minutes, and number of LTA records of 6 to 7 (~36 to 42 seconds). Note that 3 minutes of dead-time after fixed interval require to servo-positioning of antenna while starting the next grid-array

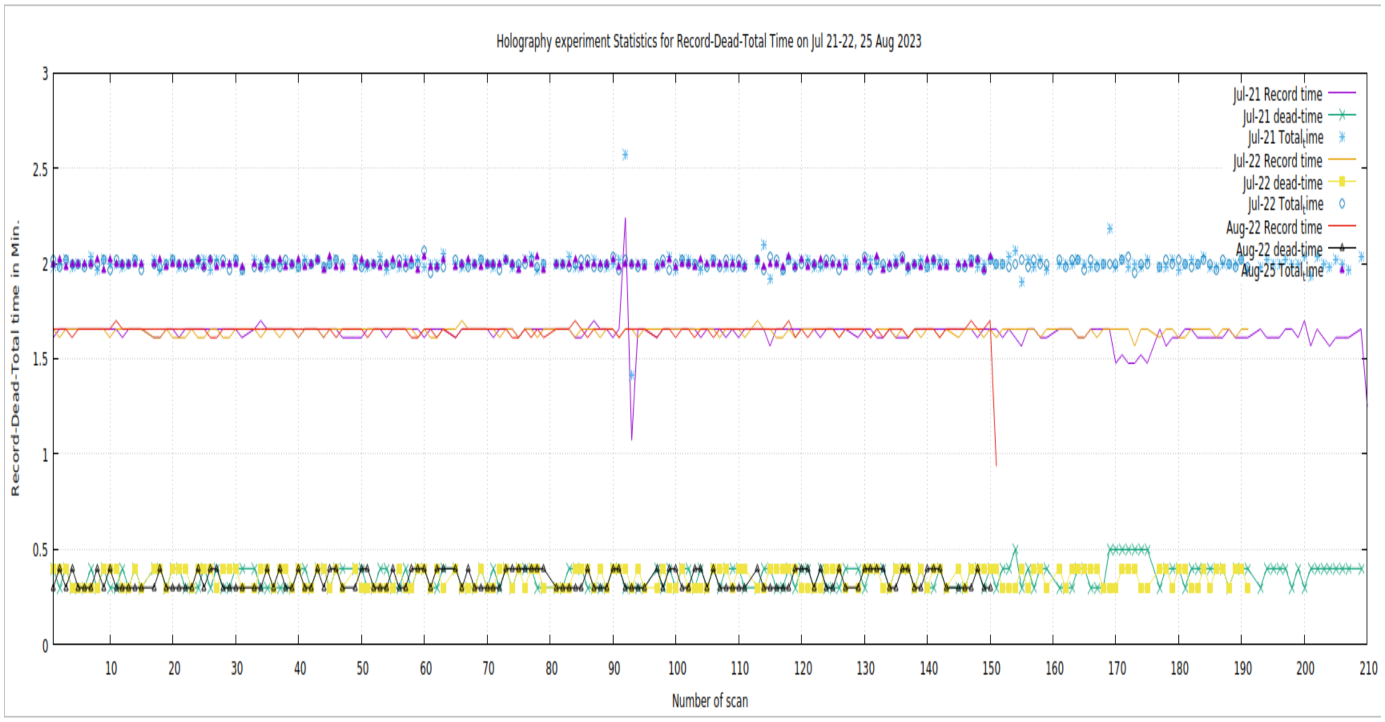


Figure 6 : Holography tests (July 21, 22, Aug 25, 2023) with 2 minutes boundry per grid-pointing. Dead-time is ~0.3 to 0.4 minutes, and recording time is ~1.6 to 1.7 minutes.

3.2 Broad-cast Command Execution for pulsar and survey kind of observations :

During the pulsar, pulsar-survey or general survey kind of observations, it is essential to cover number target-sources within a allocated time for observation. In case of pulsar survey observations, both the backends and beam-formers are used with the multiple projects (for e.g. *subar-1 and subar-2 projects for each backend*). In this case, the serial execution of the DAS-commands in the TGC system per project use to take more time with dead scan-time in range of ***~2 to 2.4 minutes of time***. Due to this number of scheduled target sources were not able to complete within a allocated time of observation using the TGC system was reported by the NCRA users.

To reduce the dead scan-time, TGC DAS-commands are implemented in broad-cast mode for both the correlator (GSB/GWB), as well as parallel execution of data-acqusitions commands for multiple subarray(s) is achived using the single command. Broad-cast method used to send data-scan command for both the correlator started/stopped data-scan simultaneously (as opposed to the time-lag of ~ 20 seconds due to serial command execution per backend). Also, time-lag between two sub-arrays for starting the data-acqusition is also reduced due to the single command usage for multiple project running under the single backend. Thus, the dead-scan time is reduced by ***~ 1 to 1.2 minutes.***

For this purpose, TGC CMC, LMC code modified, scripting APIs modified to check status of DAS-command execution for the individual project so that failure-handling can be done by retrying the DAS-command to only failed command. In the TGC code new attributes (Phase-center for each project) are introduced to track the status of each project.

For validation testing purposes, test-observation are done either using the source(s) having the same RA-DEC but with nomenclarue of source(s) differentiated by letter ‘A’ and ‘B’, or two target-source observation separated by 30’ or 5 deg in declination. Close sources selection helped in measuing the TAT and dead-scan time in realistic manner so that time-estimation for pulsar or survey kind of observation can be given. Total seven to eight tests were conducted, Table-5 gives the details of test conducted in the TGC using broad-cast execution of DAS-commands.

Mainly three kind of test-observations were done using the two backends (GSB and GWB) , and two subarrays (i.e. total four projects) :
(i) Default-mode in use - serial DAS-command execution per backend and per sub-array.

(ii) ‘fstartscan’ command in the broad-cast mode, but serially executed for subarray-1 and subaray-2.

fstart_proj('GSB,GWB',1,<source-name>)
fstart_proj('GSB,GWB',2,<source-name>)
	fast-start scan interface with **broad-cast facility** for single-subarray.

(iii) Single Broad-cast command das-command using the single command for multiple sub-arrays..

set_source_corr([1,2],'3C286,3C123','GSB,GWB')	*Add and configure the phase-center source for Subarray-1 and 2 of both GSB and LMC backend.*
strtndas('GSB,GWB','1,2')	*start the data-scan for multiple subarrays and multiple backends in one go.*
stpndas('GWB,GWB', '1,2')	*stop the data-scan for multiple subarrays and multiple backends in one go.*

TABLE – 5 : *Test observations to validate the Broad-cast and parallel execution of DAS-commands using a single command in the TGC system*

#	Date	Experiment Type	Description	Turn around time, and Dead-scan optimization time
11	**21 Jun 2023**	Multi-subarray broad-cast functional test	**Time comparison for command execution (Turn-around-time) :** **(i)** Default das-commands in serial mode for the GSB and GWB, and serial separate commands for sub-1 and sub-2. **(ii)** Default das-commands in broad-cast mode for the GSB, GWB and sub-1 and sub-2 **(iii)** Broad-cast command 'fstartscan' for the GSB and GWB but serialy for sub-1 and sub-2	***TAT is reduced by 50 % in broad-cast method.*** **(i)** For subar-1 and subar-2, the Serial execution of das-commands takes ***TAT ~ 95 seconds.*** **(ii)** In broad-cast mode, default-das commands takes ***TAT ~ 38 to 43 seconds.*** **(iii)** Using the 'fstartscan' in broadcast, ***TAT ~ 38 to 40 seconds***. **(iv)** In serial mode, stopscan for two subarrays takes ***TAT = 32 seconds.*** In Parallel mode, 'stopscan' takes ***TAT = 15 seconds*** only.
12	**05 July 2023**	**Source :** 3C147/3C147-OFF (by 5 degree) **Total scan: 7** **Recording** : 2 min per scan	**Time comparison for Time-lag for the starting project-scans between Subar-1 and Subar-2 :** **(i)** Default das-commands in serial mode for the GSB and GWB, and serial separate commands for sub-1 and sub-2. **(ii)** Default das-commands in broad-cast mode for the GSB, GWB and sub-1 and sub-2 **(iii)** Broad-cast command 'fstartscan' for the GSB and GWB but serialy for sub-1 and sub-2	**Time-lag for starting the project between subarray-1 and 2 for each correlator is** : **Default =** ~33 seconds f**startscan =** ~23 seconds **Broad-cast** = ~ 2-3 seconds. **(ii)** Dead-scan time is reduced by ~30 seconds.
13	**10 July 2023**	**Source :** 3C286/3C286-OFF (by 5 degree)	Broad-cast method, fstartscan in serial for subar 1 and subaray 2, and default (complete serial	**Dead scan-time** for **(i)** Default ~ 2 to 2.5 minutes **(ii)** Broad-cast ~ 1.3 to 1.5 minutes

		Total scan: 5 **Recording** : 2 min per scan	mode) functional testing using the GSB and GWB with subarray-1 and 2	
14	**12 July 2023**	**Source :** 3A286/3B286 (RA-DEC of 3C286) **Total scan: 20** **Recording** : 2 min per scan	Broad-cast method, fstartscan in serial for subar 1 and subaray 2, and default (complete serial mode) functional testing using the GSB and GWB with subarray-1 and 2	**(i) Total Dead time in test reduced by 20% for subar-1 and 10% for subar-2** **(ii) In default serial mode, each scan started with ~ 20 seconds separation in time.** **Whereas in Broadcost mode each scan started without any delay ~ 0 to 2 seconds only.**
15	**15 July 2023**	**Source :** 3C48A/3C48B (RA-DEC of 3C48) **Total scan: 20** **Recording** : 1.5 min per scan	**(i)** Default Serial Method – each DAS command to the backend and subarray(s) execute serially. **(ii)** 'fstartscan' command to each single subarray but in broad-cast mode for both the correlator **(iii)** Default-das command in complete parallel mode for both the backend(s) and subarrays.	**(i) TGC <u>time-delay for das-command execution to each subarray</u>** : **(a)** Default Serial mode **~ 35 (+/- 3 seconds)** **(b)** Subarray-wise 'fstrtdas' command ~ 23 seconds **(+/- 3 seconds)** **(c)** Broad-cast, parallel execution for subarray(s) **~ 0 to 3 seconds.** **(ii) <u>Dead-scan time per scan is reduced by 50 %</u> :** (**a)** Default serial mode ~ **2 min** +/- 0.3 min (**b)** '***fstartdas***' and complete '***parallel***' mode **~ 1 min** +/- 0.3 min.
16	**18 July 2023**	**Source :** 3C147A/3C147B (RA-DEC of 3C147) **Total scan: 48** **Recording** : **1.5 min per scan**	Multi-subarray (sub-1 and sub-2) , and both the backend (GSB/GWB), broad-cast command with parallel das-command execution.	Consistent ~1 minute per dead-scan time over 47 scans. Total time 170 min, with ~1.7 min recording per scan. **dead-time is** ~ **35%** and **recording 65 %**
17	**29 July 2023-01 Aug 2023**	**Source :** 3C48A/3C48B (RA-DEC of 3C48) **Total scan: 20-48** **Recording** : **1 min per scan**	Multi-subarray (sub-1 and Sub-2), and both the backend (GSB/GWB). Broad-cast command with parallel-das command execution, and sub-array wise 'fstartscan' command.	**(i)** TAT for parallel das-command execution is ~ 67 to 70 seconds **(ii)** TAT for 'fstartscan' subarraywise in broad-cast mode is ~ 68 to 70 seconds

- **Validation test results :**

(1) Dead scan-time measured in the test data of four to six experiments conducted during July 10, 12, 15, 18, and Aug 1, 2023 (***Figure 7-A and 7-B***) show that dead scan-time reduced by 50% due to the implementation of broad-cast command for both the correlator and single command usage for multi-subarray projects. This can help in effectively observing more number of targets during the pulsar survey and general survey kind of observations.
(2) The default serial DAS commands execution (old method) takes total Turn-around time ~ 2.2 to 2.9 minutes for one data-acqusition scan starting (***Figure 8-A Mar 23, 2023***) . The turn-around time measured on 3C48A-B for 20 to 48 scan blocks on July 29th, and Aug 1st, 2023 (Table 5 : ***Observation no.17***) is depicted in ***Figure 8-B*** (using broad-cast 'fstartscan' method), and ***Figure 8-C*** (using broad-cast, single command for multiple subarrays both correlator). Both the methods using broad-cast commands show that the Turn around time is reduced by 50% (i.e. ~ 1 to 1 minutes).
(3) The default serial DAS command execution previously was introducing the time-lag for starting the multiple sub-array scans due to the command execution was in serially to each correlator, and was introducing ~ 17 seconds of delay for starting the scan between to correlators (GSB and GWB). Also, due to serial execution subarray-wise, the data-acqusition starting between subarray-1 and subarray-2 for each backend was taking around ~ 35 seconds. Using the broadcast mode and parallelization using the single command, this time is reduced to less than 5 seconds for all subarrays of both the backend. ***Figure 9-A and 9-B*** shows the data plotted for tests condcuted on July 15, 2023 (***Table-5 , Observation no. 15***) using the broad-cast methods.

Figure 7-A : Dead scan-time comparison between ‘default-serial’ (***def***) mode and ‘fstartscan’ broad-cast mode (***Fstrt*** : ‘fstartscan’ command separate for each sub-array). Dead scan-time reduced from 2 to 2.3 minutes to 1 to 1.3 minutes for subarray-2 and 0.6 to 0.8 minutes for subarray-1

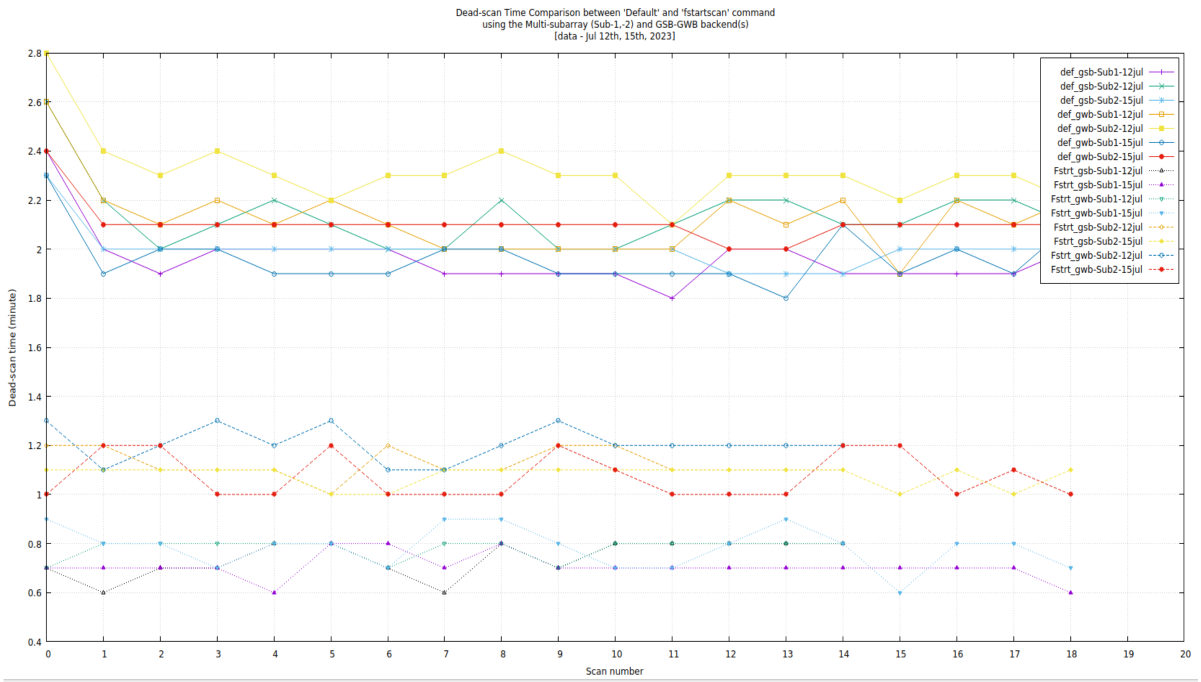


Figure 7-B : Dead scan-time comparison between ‘default-serial’ (***def***) mode and broad-cast mode using single command (***strt***) 10 Jul to 01 aug, 2023. Dead scan-time reduced from 2 to 2.3 minutes to 1 to 1.3 minutes.

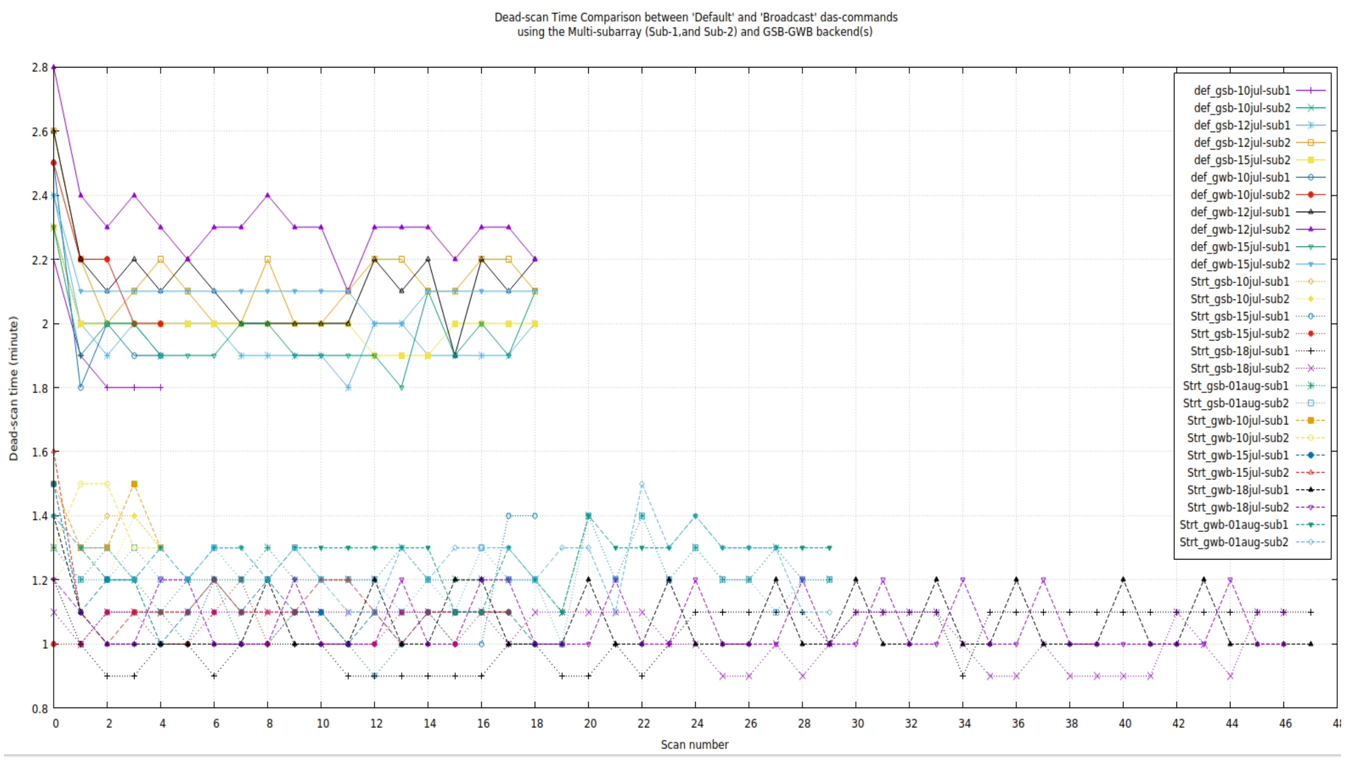


Figure 8-A : Default serial mode multi-subarray data of the GTAC obs 43_022 (Mar 21, 2023), Turn-around time is ~ 2.2 to 2.9 minutes

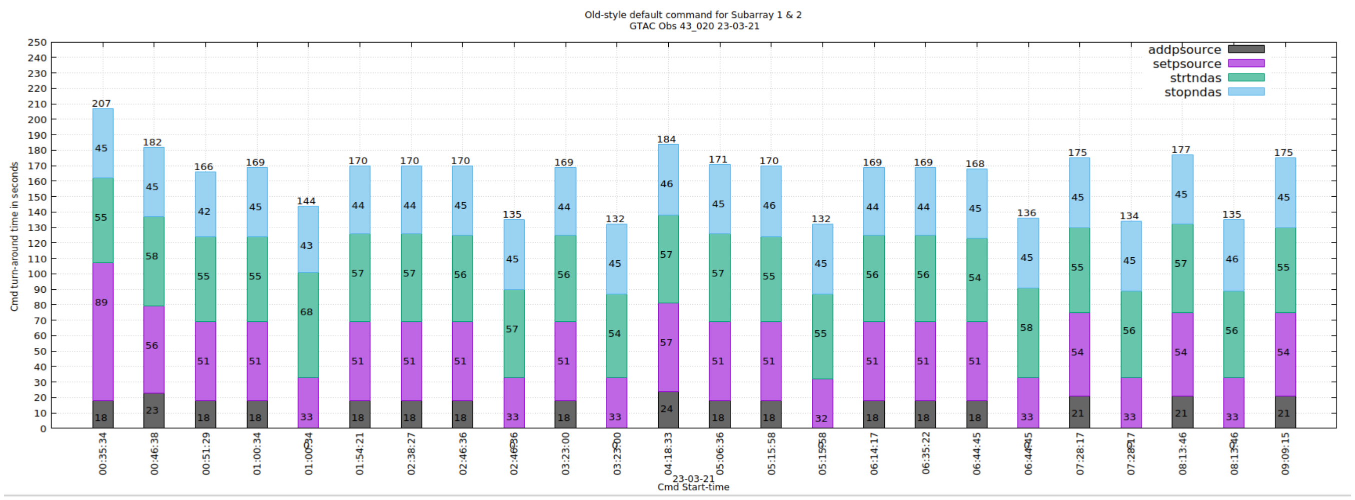


Figure 8-B : Broad-cast mode using 'fstartscan' for sub-1 & 2 (Aug 1, 2023), TAT is ~ 1 m.

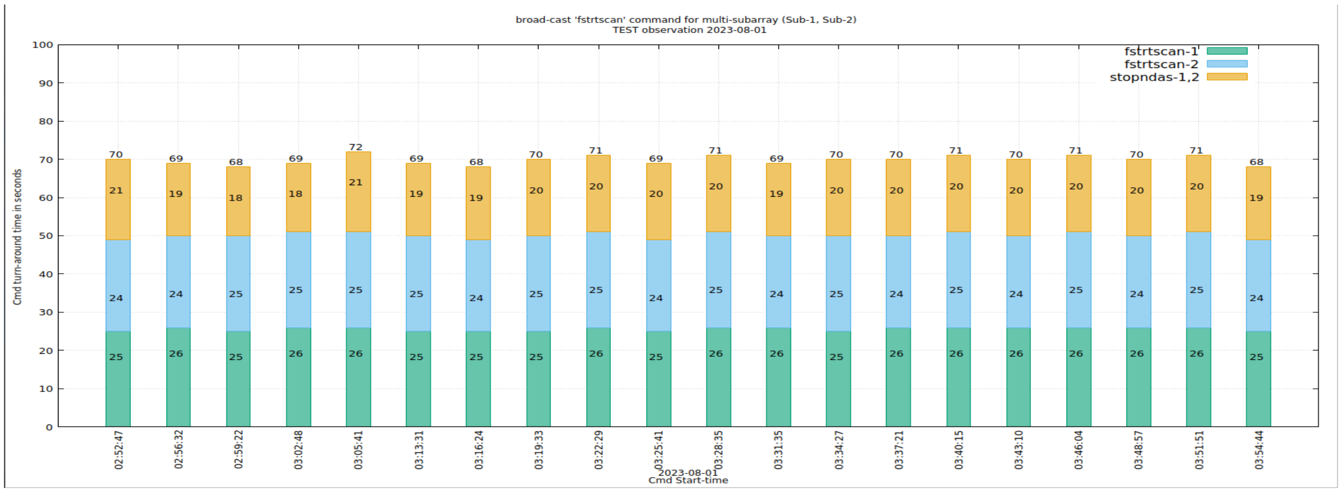


Figure 8-C : Broad-cast DAS command for multi-subarray (Jul 29, 2023), dead-scan time is ~ 1.1 m

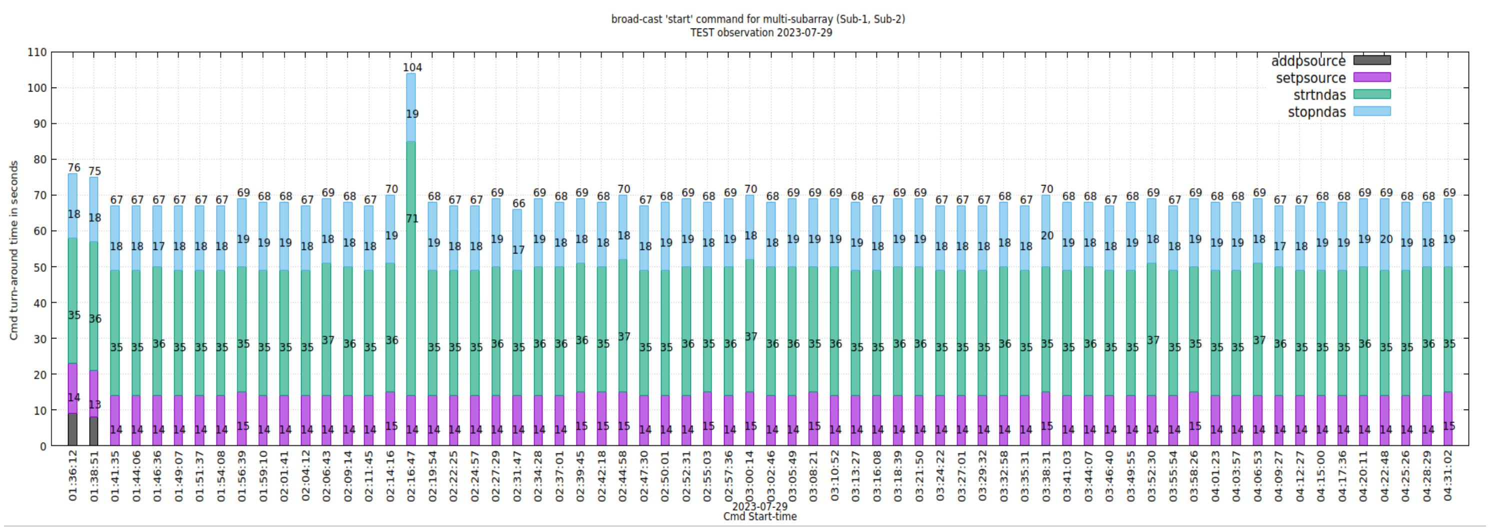


Figure 9-A : start-scan time difference between the two GSB and GWB for subarray-1 and subarray-2 broad-cast mode (***strt/fstrt***) takes less than 2 seconds as compared to default (***def***).

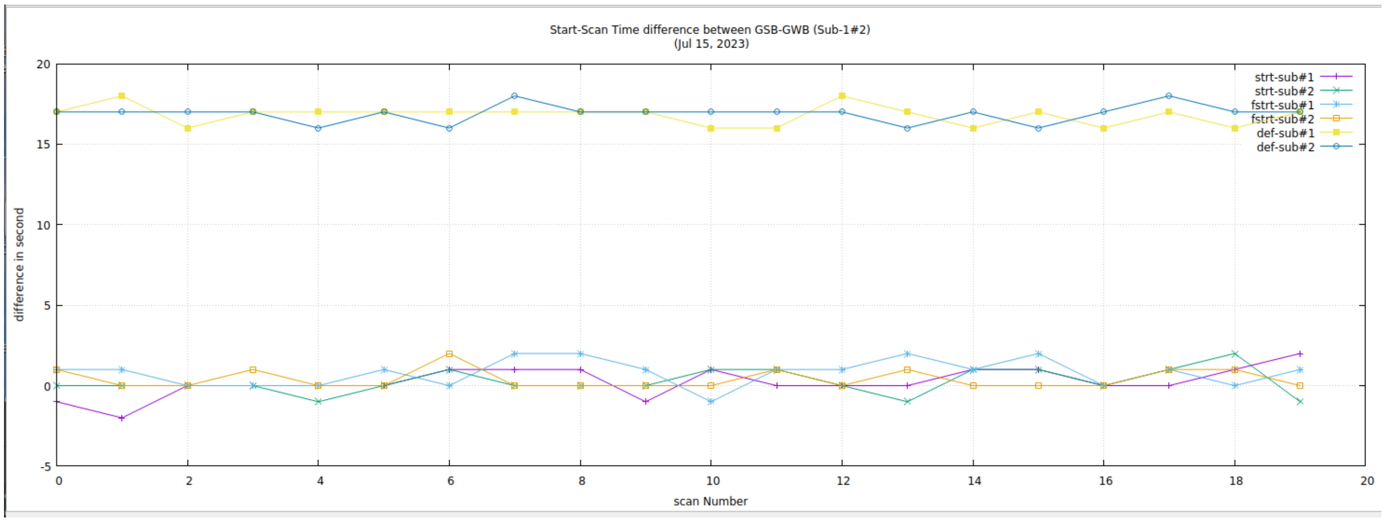
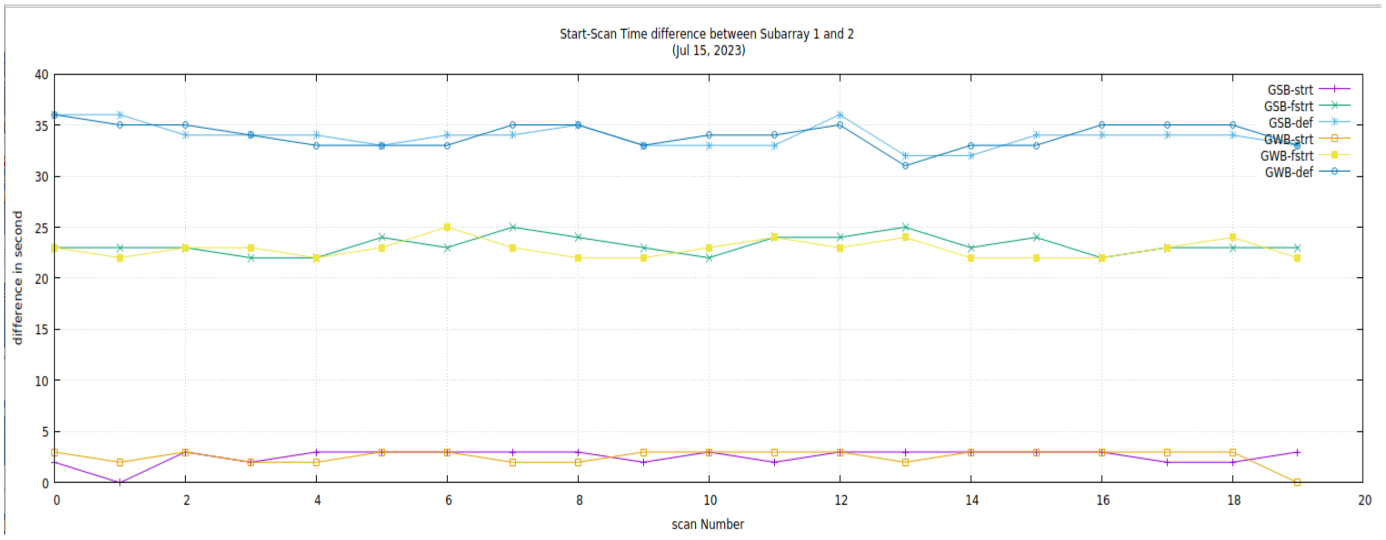


Figure 9-B : start-scan time difference between subarray 1 and 2 (Jul 15, 2023), default (***def***) ~ 35 sec, fstartsan (***fstrt***) ~ 24 seconds, broadcast (***strt***) ~ 0 to 3 seconds.



3.2.1 Broad-cast Command Execution Test using three subarrays per Backend :

The broad-cast command with multiple sub-array projects also tested using the three sub-arrays per backend (GSB and GWB). Thus total six projects were run successfully for reliability check using the single command for starting and stopping the data acqusition on Aug 10, 2023 (Refer **Table -6**).

- <u>**Validation and reliability test results**</u> **:**

(1) We could run total three subarrays per backend (GSB and GWB), with total six projects successfully where data-acqusition start and stop in the TGC system uses single command. In the reliability test described in ***Table-6***, total 10 scans for each project using the source 3C286A, 3C286B and 3C286 tried.

Testing ran successfully without failure of any DAS commands, and phase-centre were updating succesfully per project.
(2) The total average Turn-around time for the default DAS commands for three sub-arrays takes around ***~ 50 seconds***, whereas using the 'fstartscan' in broad-cast mode but separate command for the subarrays in series takes total average Turn-around time ***~ 70 seconds***. See **Figure 10-A.**
(3) The average dead scan-time using the ***'fstartscan'*** command in broad-cast mode but separate sub-array wise for three subarrays takes ***1.3 to 2 minutes*** of time. Whereas for broad-cast mode with three sub-arrays per backend i.e. total six projects takes only ***~1.1 to 1.5 minutes***. The sub-array wise dead-scan times is shown in ***Figure 10-B*** for both the methods.
(4) Thus, whenever there are multiple subarrays observation, the default DAS commands in broad-cast mode with parallel execution of sub-array commands is effective in time optimization. Therefore, 'fstartscan' command is not recommended for the multiple sub-array mode. The 'fstartscan' command shall be used in case of single sub-array observations like grid-pointing, holography, and continuum/line observation with the single sub-array.

TABLE – 6 : *Test observations to check the reliabiliy of the Broad-cast and parallel execution of DAS-commands using a single command with total three sub-arrays per backend in the TGC.*

Date	Experiment Type	Description	Turn around time, and Dead-scan optimization time
10 aug 2023	**Source :** 3C286[A,B,C] (RA-DEC of 3C286) **Total scan: 10** **Total project :** *6 (3 for each backend) Recording :* **1 min per scan**	Multi-subarray (Sub-1, Sub-2 and Sub-3) and both the backend (GSB/GWB). Broad-cast command with parallel-das command execution, and sub-array wise 'fstartscan' command.	**(A) DEAD-TIME :** **(i)** Subarray-wise 'fstartscan' in broad-cast mode : **Subar 1 : 1.1 to 1.4 min** **Subar 2 : 1.5 min** **Subar 3 : 1.8 to 2 min** **(ii)** Broad-cast command with the das-command in parallel execution : All-subarray(s) of both the **backends : 1.1 to 1.4 min** **(B) Turn-Around Time (TAT) :** **(i)** Total Subarray-wise 'fstartscan' in broad-cast mode for 3 subarrays of both the backends : **~ 93 seconds** **(ii)** Broad-cast command with the das-command in parallel execution : **~ 90 seconds** **(C) Time-delay between two subarrays (Sub3 – Sub-1, Sub2 – Sub1) :** **(i)** Total Subarray-wise 'fstartscan' in broad-cast mode

			for 3 subarrays of both the backends : **Subar 3- Subary 1 ~ 55 seconds** **Subar 2 – Subar 1 ~ 24 seconds** **(ii)** Broad-cast command with the das-command in parallel execution : **Subar 3 – Subar 1 = Subar 2 – Subar 1 ~ 2 to 5 seconds**

Figure 10-A : Average Turn-around time for three subarrays per backend (Total six projects) shows comparable time for the broad-cast commands for the ‘fstartscan’ and default das-commands with parallel execution (Data taken on Aug 10, 2023).

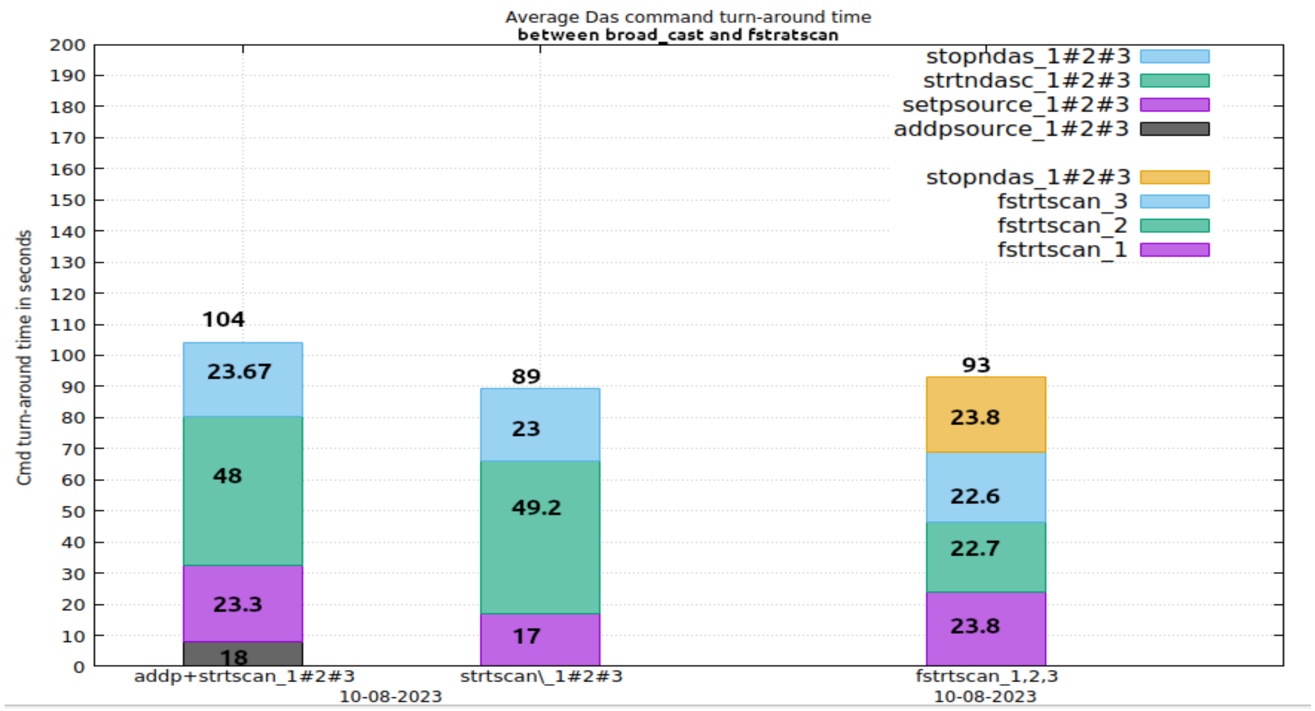
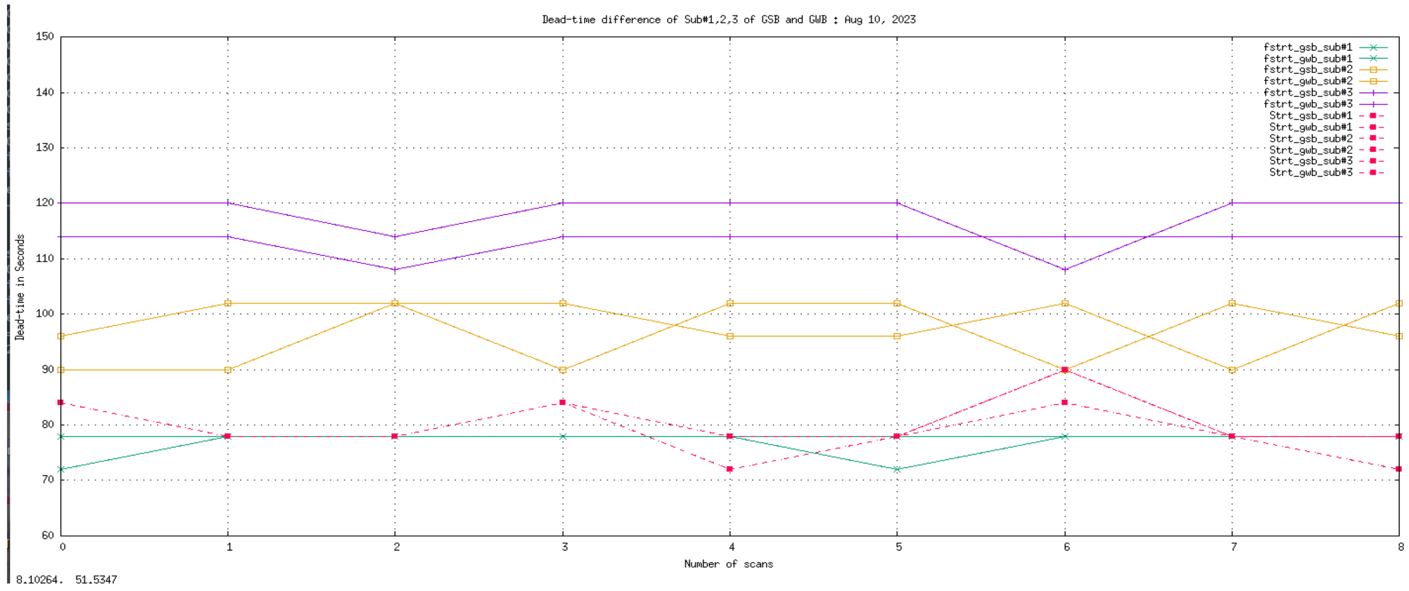


Figure 10-B : Dead scan times for three subarrays per backend (Total six projects) show that ‘fstartscan’ per subarray takes more dead-time than the single broad-cast command for three subarrays in parallel execution (Data taken on Aug 10, 2023).



4. The TGC problems resolved in dead scan-time optimization work :

(a) Target-source for the sub-array tracking was appearing wrongly as a phase-centre source. The GUI python and Java Aggregation code modified to introduce separate phase-centre dynamic attributes per backend (***PhaseCentreA*** for GSB and ***PhaseCentreB*** for GWB) and for each subarray number.

(b) During the holography experiment, if the given source name (for e.g. PN002) is same in the data-base then the 'fstartscan' command was picking up the old coordinates. The problem was occurring due to the thread concurrency issue which has been resolved by concurrency locking mechanism (Data Analysis : July 4-5, 2023 , 26 Jul 2023 confirmed that no problem reocurrence further).

(c) During the grid-pointing execution on multiple sources with diffrent HA for pointing model experiment, occasional missing scan is fixed in the Python Script-Manager by reissuing the command again if the start data-acqusition command is failed.

(d) If the multiple-subarray/Backend single-command failed for a particular sub-array , it used to re-try the command again for all the backend and subarray. This has been handled by issuing commands to only specific sub-array for which the command is failed.

(e) Some das-command was failing if the Operator-workstation ownership of the Backend (GSB or GWB) is different than the work-station in which the project is triggered. This has been resolved by reconfiguring the data-base.

(f) The antenna tracking algorithm is also optimized by internally sending a servo-system ‘***abort***’ command before issuing the new ‘*track*’ command to the servo-system. The use of ‘**abort**’ command helped in removing the stale-source coordinates of the previously tracking source in the servo-system and saved time up to ~30 seconds.

(g) The dead-time is optimized by omiting faulty antennas from the '***gotosrc()'*** waiting loop which ensures that every antenna in the subarray is tracking correct source. Previous algorithm was waiting for those antennas which were having problems such as the antenna/servo is in manual mode, servo system AZ/EL brakes are not releasing, and Servo computer malfunctioning etc.

5 Summary and Conclusion :

(i) The TGC serial implementation of the multiple DAS commands was causing the dead scan-time from 2 to 3.5 minutes. The extensive modification in the TGC code was done to implement the broad-cast DAS command, also use single command for starting the scan ('fstartscan' or single DAS-commands like 'startscan'/'stopscan' for multiple subarrays) drastically reduced the command turn-around time. Thus, the dead-scan time is reduce to ~ 1 to 1.5 minutes in case of normal observations, and 0.3 to 0.7 minutes in case of grid-pointing and holography kind of observing sessions.

(ii) In case of grid-pointing and holography experiment, compatibility of the command execution with the legacy ONLINE is required to cover number of pointings (and one minutes execution per pointing in case of holography experiment). The time optimization is achived in the TGC by implementing the single 'fstartscan' command instead of multiple das-command and algorithm changes in command sequence execution. Hence dependency on the legacy ONLINE is no more required. Also, as compared to previous multiple serial DAS command execution method, it became possible to save 1 to 1.25 hrs of time per 200 pointings of observation.

(iii) The turn-around time reduction and dead-scan time reduction is achived by 50% using the broad-cast command method and parallel execution of command. In case of multi-subarray pulsar survey or general survey kind of observation it is possible to cover observation of number of expected celestial sources.

(iv) For observations using the multiple subarrays, the default DAS commands in broad-cast mode with parallel execution for sub-array is more effective in time optimization. Therefore, 'fstartscan' command is not recommended for observations in the multiple sub-array mode. The 'fstartscan' command shall be used in case of single sub-array observations like grid-pointing, holography, and continuum/line observation.

APPENDIX-I

Detail of modifications done in the TGC code for the dead-time reduction in consecutive astronomical data scan is as follows :

#	Description	Node Type	Files and Modules
1	**New 'fstartscan' command implementation**	CMC+ LMC	**mnc_custom_db, lmc_custom_db (mysql) :** '*fstartscan*' command with 18 arguments incorporated with valid argument ranges and IO device node-association.
		CMC	**CommandFormation.java :** New module *replaceFstartscanproj*() **(i)** replaceFstartscanproj() - Method replaces the project code, antmask, get source fields for respective subarray from database in command (Feb 10, 2023) **(ii)** replaceFstartscanproj() - modified for broadcast Backend command (May 29-31, 2023)
		CMC	**CmcConstants.java :** ENTITY_[NAME, PROPERTY, OP_SHORT] attributes for the 'fstartscan' command
		CMC	**corrapi.py** : fstart_proj() API module for 'fstartscanproj' command. If the phase-centre source is new or already set then change '*op_short'* argument accordingly, broad-cast command to both the correlator, and retry the command if it is failed to one of the correlator is incorporated in the 'fstart_proj() command. **userproc.py :** 'fstrtndas' simplified command to use-at scripting terminal and in observing 'batch'.
		LMC	**CmdProcessingFunctions.java** / CmdProcessingFunctions() : source-catalog and database interface related function added (dbCustomIF) **(ii)** Based on 'Op_Short_FRQ' (frequency) argument value set (0 or 1) set fringe-stop RF-LO parameters for the correlator. Similarly argument 'Op_short_SRC' values (2,1,0) add source to project catalog,set phase-center source , or do nothing if phase-center source is already set.
2	*Broad-cast command with multiple subarrays* : **Problem :** phase center name is wrongly assigned the celestial source-name which is given for the sub-array tracking.	CMC	**PostProcessing.java :** new case for starproj, startscanproj, fstartscanproj introduced in the post-processing thread to update the 'phase-center-A/B' for the GWB and GWB (April-May 2023). Multi-subarray project implementation for the phase-center for each backend-wise subarray (Jun 16, 2023)
		CMC	**CMCConstant.java :** Phase Centre attributes 'APHASECENTRE, BPHASECENTRE' added for two types of correlator (At present GSB and GWB),
		CMC	**SkyPlot.py :** addProjectDetails() : Logic to configure PhaseCentre[A,B] is added for the GSB, GWB.
		LMC	**PostProcessing.java :** Update multiple subarray project status (running or stopped), phase-center name and dynamic attributes for 'startscanproj' and 'fstartscanproj'.
3	**Problem :** Project	CMC	**CommandBuffer.java** : run() thread to resolve project can be

	configured by non-owner of the backend (GSB/GWB) can not start or stop.		configured by any operator work-station irrespective of the Backend GSB/GWB ownership
4	**Broad-cast command with multiple subarrays**	CMC	**CommandFormation.java :** **(i)** replaceSubarrayWithPrjCode() - For each Backendwise replace each subarray number with project code define. (May 29, 2023)
		CMC	**corrapi.py :** start_proj(), stop_proj() modules modified to cater : **(i)** the broad-cast command to both the GSB and GWB and multiple subarray arguments for parallel command execution. **(ii)** Check the status of start or stop data aquisition commands, and if it is failed to one of the project among multiple projects then reissue the command again.
		CMC	**userproc.py :** User level easy to use and optimized command such as *strtndas/stpndas('BOTH' ,'1,2')*
		LMC	**commandFormation.java :** formCommandForStartProj() modified to accept given hash-separated arguments for multiple subarrays and pass the multiple arguments for 'startproj' command which is based on projects (*as opposed to subarray number, Jun14, 2023*)
		LMC	**gmrtServer[.cpp,.h]** <u>IO das-device server modifications :</u> **(i)** New logic implemented to avoid the race-condition between nonPeriodicCommand thread (Subsystem_DS) and GMRTServer threads. **(ii)** '*multipleCommandFlag*' logic developed to handle multiple-subarray command responses from the GSB or GWB device-client(s) and sending it to the parent node (CMC/AGN nodes) by copying multiple responses to form the aggregate response. **(iii)** Reporting response-status (Success / Failed) of individual command issued in parallel mode to the subarrays of each backend.
		LMC	**Subsystem.cpp** <u>IO das-device server modifications :</u> Enabled generic group-command implementation in the tango I/O device server.
		GSB/ GWB	**DeviceClient.cpp :** formDeviceResp() - Synchronise command logic changed.
5	**Asynchronous DAS-command implementation (startdas) for the VLBI**	CMC/ LMC	**mnc_custom_db/lmc_custom_db (mysql) :** modified to incorporate ***time_str*** argument for the command '*startproj*', '*startscanproj*' , '*startscanbeam*', and '*fstartscanproj*'.
		CMC	**CommandFormation.java:** *replaceSubarrayWithProjCode*() - incorproated time_str argument in the 'startscan' command.
		LMC	**CommandFormation.java:** *formCommandForstartproj*() formCommandForstartBeam : time_str argument send to the GSB/GWB deviceclient.

			LMCConstant.java : Introduced new variable for expected time argument (time_str)
		CMC	**corrapy.py** : start_proj, fstart_proj() module modified to take optional ***time_str*** argument in HH:MM:SS to trigger the astronomical data scan at expected time.
		GSB/ GWB	**DeviceClient.cpp : (i)** *checkBufExec*() Check previous project command time threshold (difference between stored trigger time and current-time) is <= 0 then execute the command, otherwise store the command with expected time (time_str argument). **(ii)** Command synchronisation logic changed in formDeviceResponse() module to handle multiple-subarray commands.