GEO Online Detector Characterization System

R. Balasubramanian

Cardiff University

LSC March 2003

LIGO-G030178-00-Z

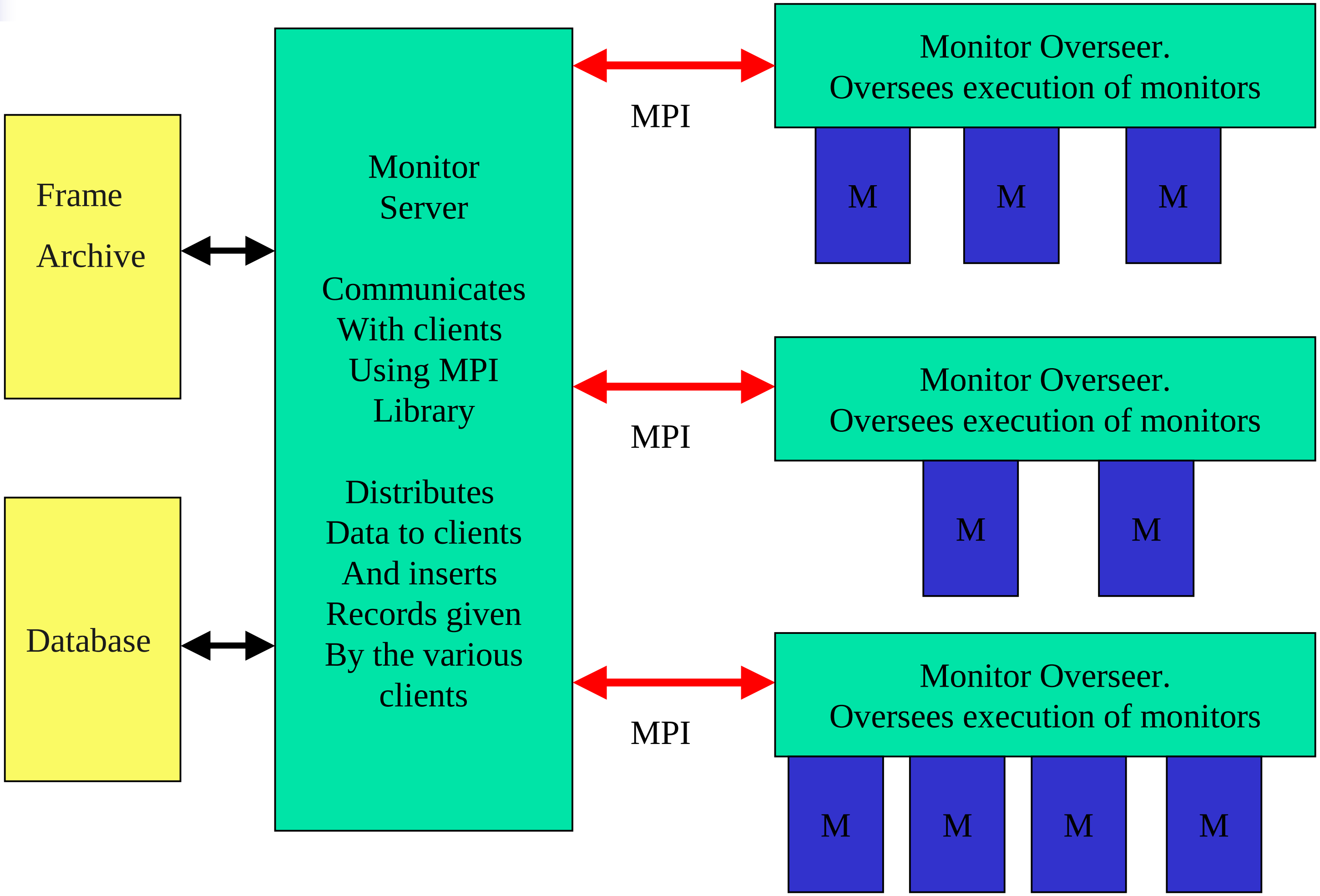
GEO Online Detector Characterization System

- The basic purpose of the online detector characterization system is to monitor and characterize the various channels produced by the interferometer.
- The software is written in C++ and includes a signal processing library
- The software includes Frame Data I/O and database I/O
- The system is designed to be highly extensible. All analysis is carried out by *Monitors* and users can determine which monitors they wish to execute.
- The system produces results primarily in the form of database records.
- Monitors can be distributed over a Beowulf Cluster.

People Involved

- Main Development Group
  - Stas Babak
  - R.Balasubramanian
  - David Churches
- Validation and monitors
  - Karsten Koetter
  - Uta Weiland
  - Martin Hewitson
  - Siong Heng

GODCS ARCHITECTURE



Frame

Archive

Database

Monitor
Server

Communicates
With clients
Using MPI
Library

Distributes
Data to clients
And inserts
Records given
By the various
clients

MPI

Monitor Overseer.
Oversees execution of monitors

MPI

Monitor Overseer.
Oversees execution of monitors

MPI

Monitor Overseer.
Oversees execution of monitors

GODCS Configuration File

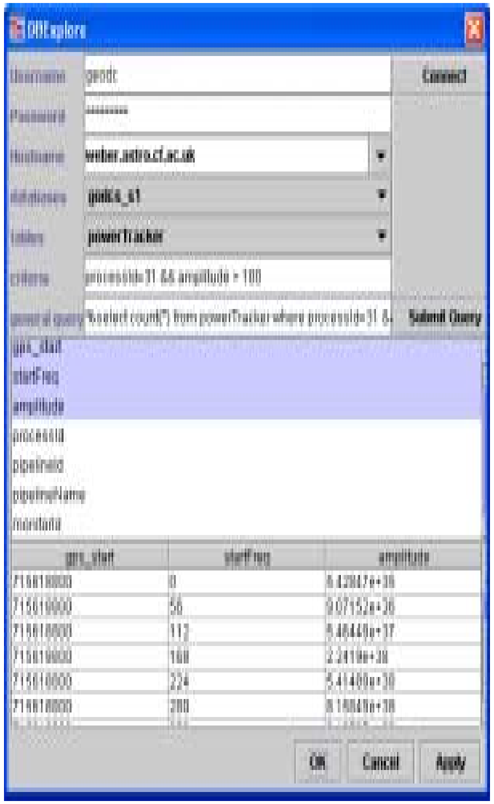
```
multiDCPipeline              !This keyword has to be either singleDCPipeline or multiDCPipeline
database(godcs)              !Give the name of the database here.
hostname(localhost)          !The host where the server is operating from
user(godcs)                  !the username
password(godcspass)          !the password
comment(InspiralMon)
logdir(/home/bala/geoda/GEOPP/bin) !The directory where the log files will be written
basedir(/data/geodata)                        !The base directory of the frame data
frameMode(skip)                    !This keyword can be wait or skip. "wait"  for online
interval(728654430 728654464)      !The gps start and end times

job(4 1 0)
monitor() FrameChannelSummaryMon() channels(G1:PEM_CBCTR_RK-C-V2) store(true)
monitor() PsdMon(1 0) channels(G1:PEM_CBCTR_RK-C-V2) store(true)
monitor() PowerTrackerMon(noise_floor 32 0) channels(G1:PEM_CBCTR_RK-C-V2) store(true)
monitor() TFClusterMon(2048 1792 30 10 .1 1)  channels(G1:LSC_MID_EP-P_HP) store(true)

job(4 1 0)
monitor() FrameChannelSummaryMon() channels(G1:DER_H_HP-EP) store(true)
monitor() PsdMon(1 0) channels(G1:DER_H_HP-EP) store(true)
monitor() PowerTrackerMon(noise_floor 32 0) channels(G1:DER_H_HP-EP) store(true)
monitor() InspiralMon(1 0) channels(G1:DER_H_HP-EP) store(true)
monitor() TFClusterMon(2048 1792 30 10 .1)  channels(G1:DER_H_HP-EP) store(true)
```

Database

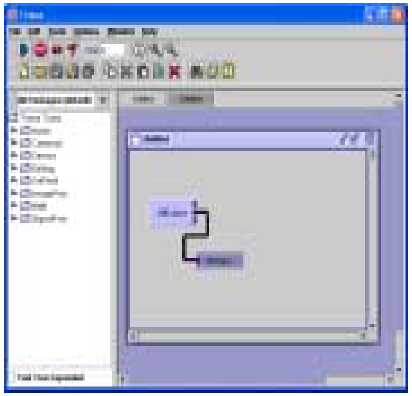
- The results are stored in a MYSQL database which is a public domain software
- Found to be reliable and fast and can handle large event rates
- Primary data mining and data exploration tool is intended to be TRIANA a java based quick look data analysis software developed within GEO.

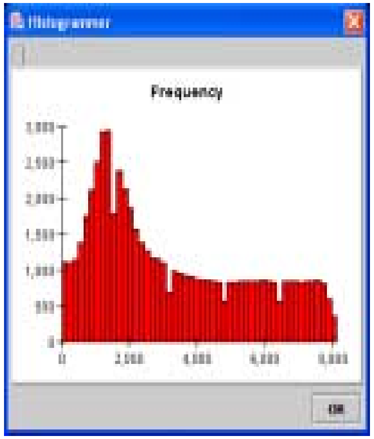


Username: geodc
Password: ********
Hostname: weber.astro.cf.ac.uk
databases: gidb_s1
tables: powerTracker
criteria: processId=31 && amplitude > 100
general query: %select count(*) from powerTracker where processId=31 &

gps_start
startFreq
amplitude
processId
pipelineId
pipelineName
monitorId

gps_start	startFreq	amplitude
715610000	0	6.42887e+36
715610000	56	9.07152e+36
715610000	112	5.46440e+37
715610000	168	2.2419e+38
715610000	224	5.41480e+38
715610000	280	8.19846e+38





Status of Project

- Serious development started beginning of 2002
- The code was online during S1 for nearly the entire duration.
- Code runs on an Alpha Beowulf cluster
- The only problems were code crashes due to faulty frame library (fixed in later versions)
- MYSQL Database software was found to be very reliable and fast.
- Validation against MATLAB in progress
- Current focus is to expand on the number of monitors and increase their functionality.
- DSP facilities adequate but will mature further.

Monitors

- Event Monitors
  - LockStatusMon
  - SaturationMon
  - InspiralMon
  - PowerTrackerMon
  - GlitchMon
  - GlitchInPowerLineMon
  - PowerLineMon
  - LineMon
  - TFClusterMon
  - InspiralSearchMon
  - CalibrateChannelMon

- Subsidary monitors
  - NoiseGeneratorMon
  - IIRBandFilterMon
  - FIRBandFilterMon
  - PsdMon
  - SpectrogramMon
  - TSpecDisplayMon

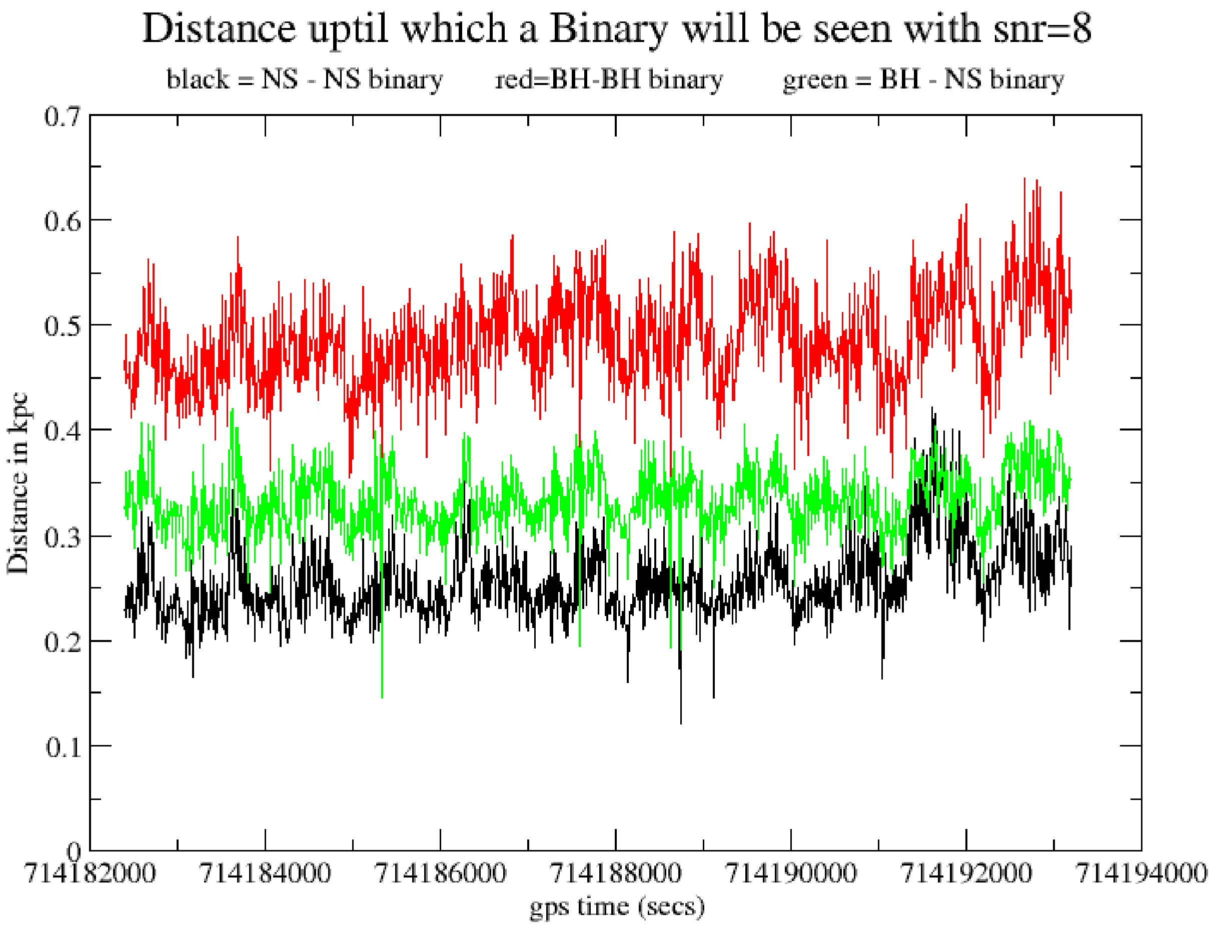
Monitor Descriptions

- LockStatusMon
  - Stores change points in the lock status of the interferometer
- SaturationMon
  - Checks to see if any of the channels saturates the ADC.
- GlitchMon
  - Looks for glitches in the time series
- GlitchInPowerLineMon
  - Looks for Glitch in the power line channels
  - Found that the UPS output did have Glitches. Investigation is still in progress to determine its influence in the h(t) channel.
- PowerLineTrackerMon
  - Tracks drifts in amplitude and frequency of the power line harmonics

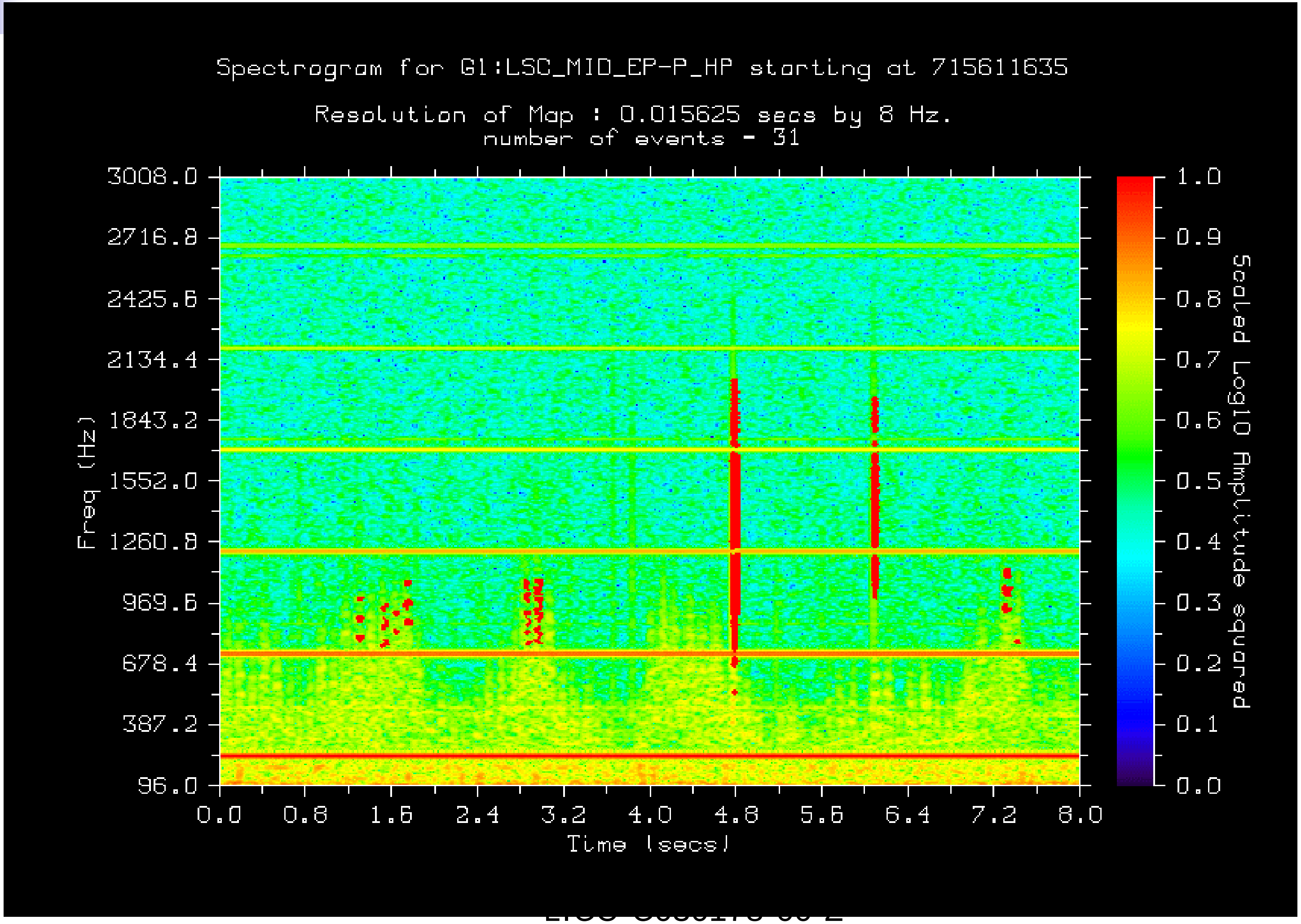
Monitor Descriptions

- PowerTrackerMon
  - Stores band limited RMS averages
  - Stores only change points in the database to reduce the number of events
- LineMon
  - Queries a database of observed lines and tracks these lines
  - Records change points in measured frequency and amplitude and stores them in the database.
- InspiralSearchMon
  - Filters the data through a small template bank
  - The filtering is mostly done with LAL code
- CalibrateChannelMon
- NoiseGeneratorMon
- PsdMon

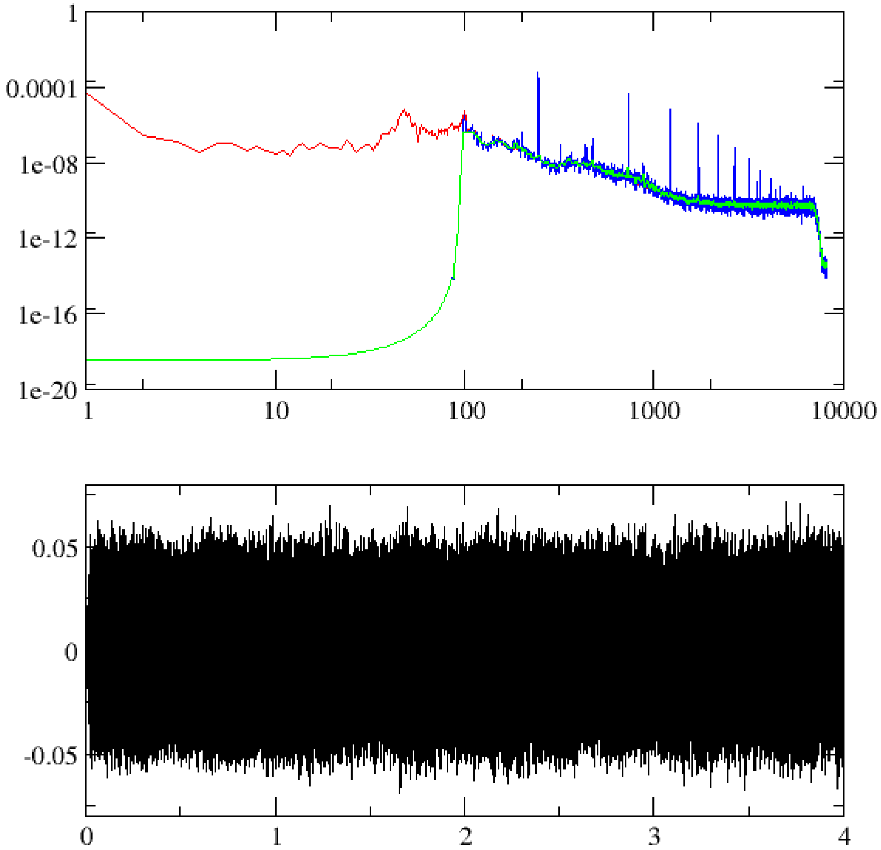
InspiralMon



TFClusterMon



TSpecDisplayMon


1
0.0001
1e-08
1e-12
1e-16
1e-20
1
10
100
1000
10000
0.05
0
-0.05
0
1
2
3
4