INTRODUCTION/OVERVIEW

The GEO++ project aims to develop software for processing large amounts of time series data for the purpose of detector characterization. In practice a GEO++ monitor is functionally equivalent to a DMT monitor, though there is no real relationship between GEO++ and DMT.

The web pages for GEO++ are at


ACCESSIBILITY/DEVELOPMENT/CVS/VERSION CONTROL

GEO++ is freely available and does not depend on any commercial packages.

GEO++ is developed in C++. Development is coordinated using a CVS archive located at Cardiff on the machine mini.astro.cf.ac.uk. Read and write access is limited to developers with accounts on mini and in the group 'geopp', and is conducted via SSH.

The developers of GEO++ are Stanislav Babak, Dave Churches, Thomas Cokelaer, and R. Balasubramanian. No CVS tagging has been done so far and there have been no formal releases of GEO++ at this time. During the GEO600 S1 and S3 runs the most recent versions available at the time were downloaded and used.

As already noted GEO++ is developed in C++. It takes full advantage of many of the features available in C++ including inheritance, operator function overloading, and it makes extensive use of the C++ standard template library (STL). At this time exceptions are not used. The entire code is developed in the namespace GEOPP.

GEO++ leverages many external libraries including the Gnu Scientific Library (GSL), FFTW, MySQL client, FrameCPP, LAL (optional), a graphics library named dislin (optional), and LAM (optional if not using MPI).

PLATFORM SUPPORT

GEO++ is compiled with gcc and has been tested with versions 2.95.3 to gcc 3.2. GEO++ has been installed on Intel RedHat Linux version 7.1, 7.3, 9.0, and on Alpha RedHat 6.2.

LEVERAGING EXISTING CODE BASES

A GEO++ program or monitor can link to LAL or other C libraries easily since C is a subset of C++, and this has already been demonstrated a number of times. Taking existing routines in GEO++ and moving them back into LAL, however, would be very difficult and time intensive because of the entanglement of data structures and methods one expects in C++ code.

The basic execution unit of GEO++ is a monitor. One selects at least
one monitor to execute, however in most cases a chain of monitors is used. For example the first monitor may be for preprocessing the data and the second monitor might for example look for glitches. A chain of monitors is called a pipeline. To run a pipeline one needs to prepare a ascii jobfile that details which monitors to run and with what arguments.

The monitors or pipelines can be either run as single processor (SinglePipeline) jobs or as MPI (MPIPipeline) jobs. When run as single processor offline jobs Condor has been used to distribute the jobs over serveral nodes in a cluster. This is true for both the InspiralSearch pipeline as well as for HACR (a GEO++ implementation of TFClusters).

MPIPipeline jobs are typically used in detector characterization, where each compute node asks for a subset of channels and run several monitors in a sequence on the channels so that the data is read only once and many monitors have access to the data.

DOCUMENTATION

Documentation for GEO++ is largely included in the header files nd extracted with doxygen. The documentation has not kept pace with the software development and this is being remedied currently.

BUG REPORTING

There is no bug reporting mechanism at this time for GEO++.

VALIDATION

Validation of GEO++ code as this time has been done by comparing results/outputs against similar from eithe MATLAB or some other standard, and then recorded in a lab book.