Document Type	DCC Number	Date
Technical Note	**LIGO-T080136-v2**	Sept 29, 2010

aLIGO CDS Real-time Software SysAdmin Guide

R. Bork, K, Thorne

California Institute of Technology
LIGO Project – MS 18-34
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project – NW 22-295
Cambridge, MA 01239
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

www: http://www.ligo.caltech.edu/

Table of Contents

1 Introduction

This document provides information on the configuration and maintenance of the software on an Advanced LIGO (aLIGO) Control and Data System (CDS) real-time data acquisition system. The scope is limited to information necessary to install and operate a generic version of such a system. The details for existing installations of such systems are in documentation maintained at each LIGO site.

2 Document Overview

During the construction and operation of initial LIGO (iLIGO), a variety of software was developed for control and data acquisition of real-time systems. For Advanced LIGO (aLIGO), much of this software will remain, as is, or, as in the case of the real-time software, has been made more modular for reuse. For the latter, a generic Real-time Code Generator (RCG) has been developed, that uses MATLAB's Simulink as a graphical user interface (GUI) for rapid development and documentation of software that is compatible with the various Data Acquisition (DAQ) and Global Diagnostic System (GDS) software.
In the following sections, the basic system requirements and installation instructions are provided to run the CDS software. This includes:

- References: Lists references to other aLIGO CDS system documentation.
- System Requirements: Lists the minimal system hardware and software requirements to operate the system.
- Hardware Configuration, including supported I/O modules
- Software Configuration, covering primarily the supporting DAQ and GDS functions.

3 References

Reference A: LIGO-T080135 AdvLIGO Realtime Code Generator (RCG) Application Developer's Guide
Reference B: LIGO-T0900602 AdvLIGO CDS Design Overview
Reference C: LIGO-T1000248 aLIGO CDS File System Directories
Reference D: LIGO-1000379 CDS Environment Configuration Scripts

4 System Requirements

The aLIGO CDS is designed to operate in a number of hardware configurations:

- A stand-alone computer, with embedded I/O modules.
- A stand-alone computer with one, or more, I/O expansion chassis to house I/O modules. The aLIGO CDS supports PCI-X and PCI Express (PCIe) modules. These modules may be inserted into an expansion chassis (LIGO custom or commercial) and accessed remotely from the host at up to 15m (cable), or 300m (fiber optics).
- A distributed control configuration, which includes multiple computers, with multiple I/O chassis, all interconnected via Ethernet and real-time networking.

4.1 System Configuration Options

4.1.1 Stand-alone System

The simplest stand-alone system has a single computer with one or more I/O modules plugged into its local PCI-X or PCIe bus. Both the real-time models, data acquisition software and user applications run on the one computer. This requires additional equipment to provide clocks for the I/O modules.

The typical stand-alone system used on the aLIGO test stands has a computer connected to an I/O expansion chassis. In the expansion chassis are the I/O modules as well as a timing slave to clock the I/O modules. For systems with a number of users, a second computer can be added from which user applications can be run without interfering with the real-time models and data acquisition.

4.1.2 Distributed System

In distributed systems, separate computers run the real-time models and the data acquisition software. The computers running the real-time models (termed front-end (FE) computers) are booted from an FE boot server. There are separate networks to carry real-time control data, DAQ data and normal Ethernet LAN traffic. The DAQ system (frame builder) has multiple computers that get the data from the front-ends (data concentrators), write that data to frame files (frame writers) and serve the data to user applications (network data servers).

4.2 Hardware Requirements

4.2.1 Real-time Computer Hardware

The CDS real-time software requires a specific computing platform, with the minimal computer requirements listed in the following subsections. Various supporting software tools run on various platforms, only dependent on the operating system installed. The operating system requirements for these are given in the software requirements section.

4.2.1.1 Processor

The CDS real-time software is designed to run on a multiple-core Intel CPU based computer. This may be a single processor with multiple cores or multiple processors with single or multiple cores. The particular clocking speed of the processors is dependent on the application needs. Typically, present systems use a minimum clock speed of 2.6 GHz.

4.2.1.2 Memory

A minimum of 2GByte RAM is required. Stand-alone systems that also run user applications (such as MATLAB) will need additional RAM.

4.2.1.3 I/O Slots

The computer must have sufficient PCI-X or PCIe slots to either house the supported I/O modules directly, or to install the PCIe modules which interface to I/O expansion chassis.

4.2.1.4 Disk Drives

The amount of disk space required is dependent on the application. For a FE computer that does not write data and is booted off a boot server, no disk is needed. For an FE boot server and some DAQ computers, a minimal drive(typically 75GB) is sufficient. For computers that acquire and write data to disk, space is dependent on the anticipated data load and amount of look back time desired.

4.2.2 Networking

4.2.2.1 Ethernet Networks

In aLIGO, we use Ethernet networks both for standard LAN access and for DAQ readout and data distribution.

4.2.2.1.1 FE (EPICS) LAN

Each computer requires a standard Ethernet port for connecting to other computers on the local network for NFS mounting of disks, SSH access, etc. For stand-alone test stand systems, a NAT router is used to provide access to external networks such as the Internet for code updates, etc. This is typically at least 100MbE (100 Megabit) but can be 1GbE (1Gigabit or 1000 Megabit). This network is also used to exchange EPICS traffic.

The FE LAN connection for FE computers at the end stations is a local Ethernet switch. That switch is then connected over fiber to the main FE LAN switch.

4.2.2.1.2 DAQ Broadcast NET

On a distributed system, the data concentrator broadcasts data to the other frame builder computers (frame writers, NDS servers, GDS) over the DAQ Broadcast NET. Currently an Ethernet switch is used that provides 10GbE connections over fiber for all the frame builder computers. This replaces the Infiniband network using in eLIGO for DAQ broadcasts.

4.2.2.2 Real-time Networks

The front-end models need to exchange control data in real-time. Real-time networks using a reflective memory (RFM) paradigm are used. They also send data to the data concentrator in real-time.

4.2.2.2.1 FE PCI NET

In a distributed system, co-located FE computers can exchange real-time data over the FE PCI NET. Currently, a Dolphin PCIe switch is used to implement a very fast reflective memory network by connecting each computer on the same PCIe bus. This replaces the GE Fanuc RFM network used in iLIGO.

4.2.2.2.2 FE RFM NET

To communicate over long distances, the real-time data will still use a GE Fanuc RFM network. The network will run from computers at each end station to a FE computer in the corner station. RFM switches will be also used.

4.2.2.2.3 FE DAQ NET

On a distributed system, the FE computers send data to the data concentrators over the FE DAQ NET. Currently an Ethernet switch is used that provides 1GbE speed to each front-end, but 10GbE over fiber for the data concentrator. This replaces the older GE Fanuc Reflective Memory (RFM) network in iLIGO and the Myricom network in Enhanced LIGO (eLIGO) systems.

Front-end computers at the end stations have a local 1GbE switch that is connected over a fiber connection to an additional 1GbE port on the main FE DAQ NET switch.

4.2.3 I/O Buses

The RCG software supports PCI/PCIe modules. These modules may either be housed in the local computer or, via a PCIe to PCIe link, to a PCI-X/PCIe expansion chassis.
To date, two expansion chassis have been used and tested, one for PCI-X modules and one for PCIe modules. Both provide for copper cable connections to the host at up to 15m and up to 300m via multi-mode fiber.

• PCIe host to PCI-X expansion chassis
• PCIe host to PCIe expansion chassis

4.2.4 I/O Modules

The present CDS software supports a limited number of ADC, DAC and binary I/O modules. The primary modules supported by the current release are listed in the following subsections.

4.2.4.1 ADC Modules

Software support is provided for the following ADC modules:
• General Standards PMC66-16AI64SSA: 64 single-ended / 32 differential 16 bit ADCs on a single PCI/PCIe module.
• General Standards PMC66-18AI32SSC1M: 32 individual SAR, 18 bit ADCs on a single PCI/PCIe module.

4.2.4.2 DAC Modules

Software support is provided for the following DAC modules:

• General Standards PMC66-16AO16 : 16 DACs on a single PCI/PCIe module

4.2.4.3 Combination ADC/DAC Modules

• General Standards PMC66-16AISS8AO4

4.2.4.4 Binary I/O Modules

Three binary I/O modules are presently supported:
• Access I/O Relay Modules
• Contec Optically Isolated Binary Output Module

4.2.4.5 Reflected Memory (RFM) Modules

For low latency, point-to-point communications between computers, the RCG software supports GE Fanuc Reflected Memory NIC.

4.2.4.6 Supporting Software Decimation / Up Sample Filter Software

By default, the CDS software sets the sampling rate of ADC/DAC modules to 65536 Hz, then down-samples/ up-samples data to the particular process rate, as defined by the user application. To accommodate this, digital filters have been designed for use with aLIGO hardware Anti-aliasing (AA) and Anti-Imaging (AI) hardware filters for aLIGO common operating rates of 32668, 16384 and 2048 samples/sec. Foton plots of the transfer functions are shown in the following figure.

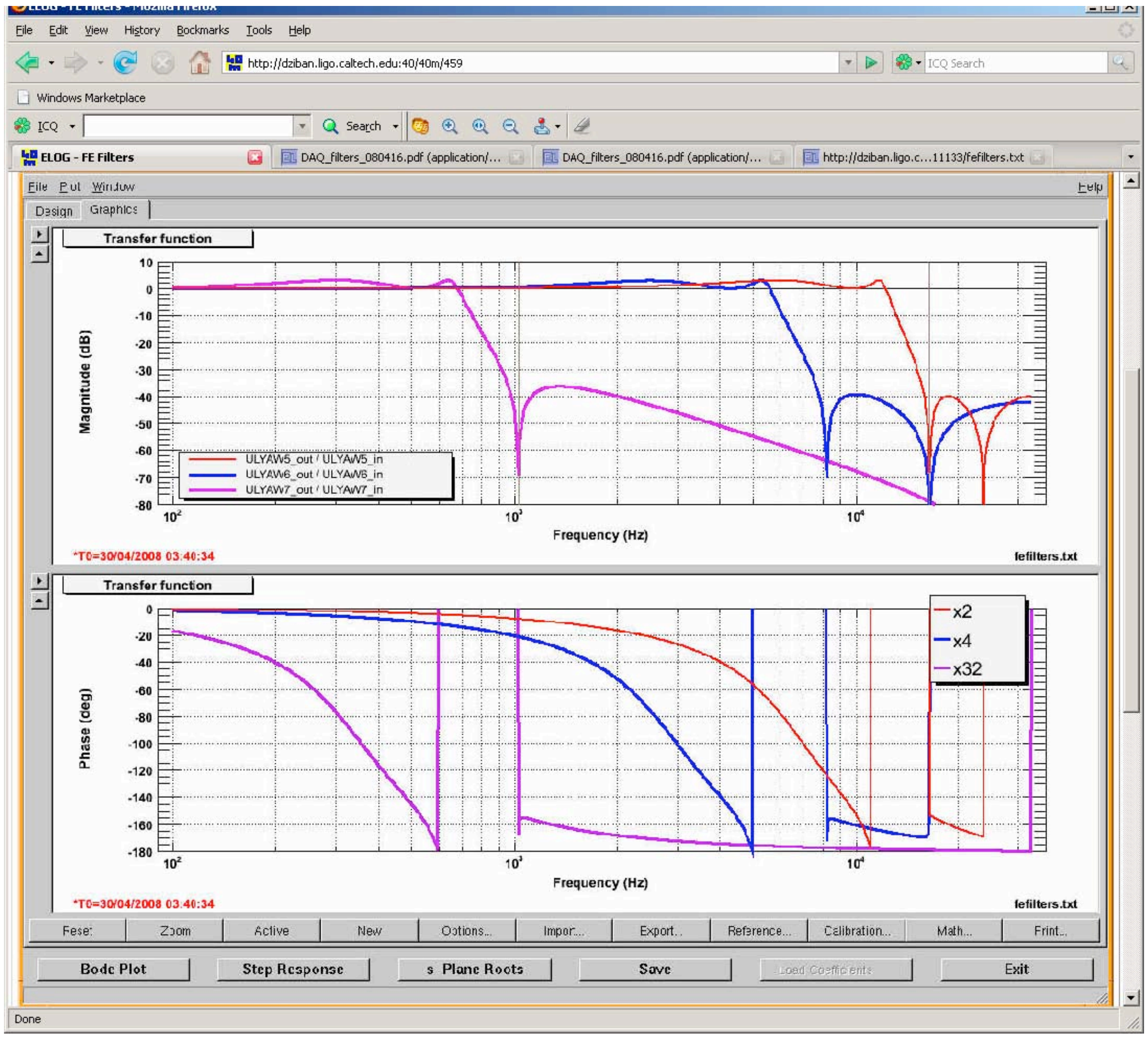

Figure 1 Decimation/Upsample Filter Response

4.2.5 Timing System

All ADC and DAC I/O modules require a square wave clock input signal. This timing signal must be a multiple of 2 kHz, in a range from 2048 Hz to 65536 Hz. This typically requires, at minimum, a single aLIGO timing slave unit (stand-alone, asynchronous system), or a timing master and timing slave(s) for a GPS synchronized, distributed system. The I/O Expansion chassis use a Timing DAQ interface to connect the Timing Slave output to the ADC/DAC cards.

- LIGO E090002 Timing Master/Fanout Module
- LIGO E090001 Timing Slave Module
- LIGO E090004 Timing DAQ interface
- LIGO E080541 AdvLIGO Optical Timing System Quick Start Guide

4.3 Generic System Software Requirements

This covers the non-LIGO supplied software needed for real-time systems.

4.3.1 Operating Systems for Real-time Computers

The real-time software is designed to run on the Linux OS. One of the following operating systems is required:

- Gentoo with LIGO CPU Shutdown patch: This is used on FE computers running real-time models, and replaces RTLinuxPro™.
- RTLinuxPro™, now a product of WindRiver®: Used on eLIGO systems and early aLIGO stand-alone systems.

For compilation, the standard gnu compilers are used.

4.3.2 Operating Systems for DAQ Computers

Distributed systems have separate DAQ computers. The DAQ computers used in the frame builder do not have to run real-time operating systems. All eLIGO and aLIGO DAQ systems use Linux for most computers, except for that hosting the Sun QFS file system used for storing frames for retrieval by LDAS. For Linux, the following are supported:

- Gentoo: This provides commonality with the aLIGO FE computers.
- CentOS: This is used on the eLIGO DAQ computers and is also supported by other LIGO software groups.

For the DAQ computer hosting the SUN QFS file system (which may also be the frame writer), Solaris 10 is currently required.

4.3.3 Operating Systems for Non-real-time Computers

For CDS supporting systems, such as operator workstations, both Solaris and Linux are currently supported. The software for these has been tested on Solaris 10 and Linux Fedora, gentoo, and CentOS. For new systems, a Linux OS is strongly encouraged.

4.3.4 MATLAB

A licensed copy of MATLAB with Simulink is required to produce the .mdl file needed by the RCG. The copy of MATLAB must be R2006b or later.

A copy of MATLAB is also required to run additional LIGO CDS tools, such as mDV and ligoDV. In addition to MATLAB, these tools require a CDS NDS MATLAB mex file to obtain data from CDS NDS servers.

4.3.5 Myrinet Express

The CDS software uses Myrinet Express for all NIC adapters on the FE DAQ NET. There is an open-source implementation (Open-MX) available at http://open-mx.gforge.inria.fr/ .

4.3.6 10GbE NIC Drivers

For distributed system hardware that has Myricom 10GbE adapters, you will need to get drivers from the Myricom web site (http://www.myri.com). For other 10GbE adapters, such as those from Sun, you will have to get operating-system appropriate drivers from the vendor

5 Hardware Configuration

5.1 Timing System

5.2 I/O Expansion Chassis

5.3 I/O Modules

Software support is provided for the following PCI modules. Note that the present ADC/DAC modules can either be purchased as either PCI-X or PCIe.

6 Software Installation and Configuration

The basic instructions for compiling, installing and running the real-time applications within the CDS environment are covered in Reference A. This section covers the computer system parameters that must be set up prior to running real-time applications, along with supporting DAQ and GDS software.

6.1 System Configuration Options

The CDS system may be set up standalone or as a distributed system. The software structure for both options is shown in the following sections to provide a brief overview. More detail is available in Reference B.

6.1.1 Standalone Code Modules and Data Flow

The following figure shows the basic software modules which run on a standalone real-time system, with a basic description following.

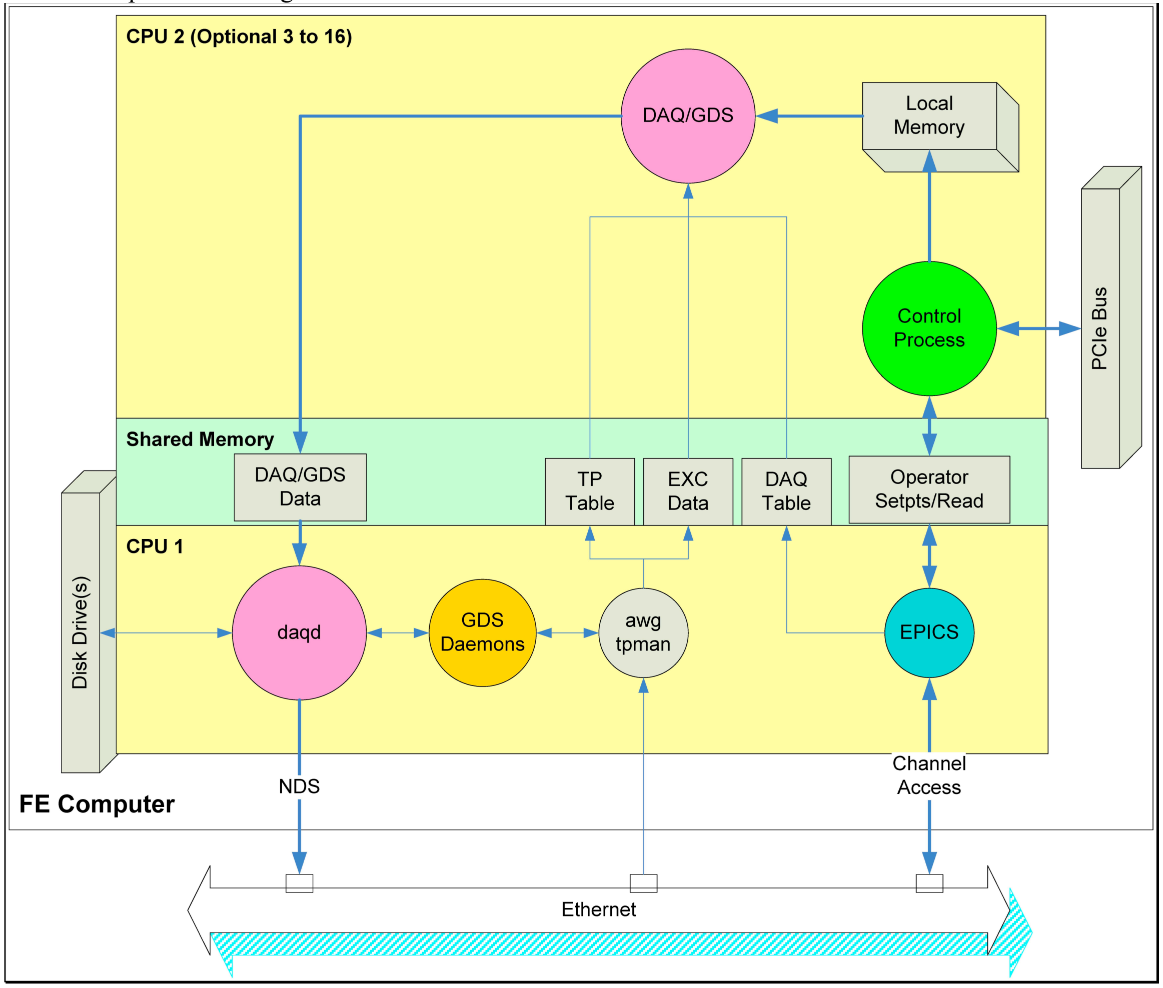

Figure 2: Standalone System Software

The non-real-time tasks always run on the lowest numbered CPU, typically CPU core 0, under the control of the standard Linux scheduler. These tasks communicate with real-time tasks via shared memory block. These tasks are:

- EPICS: Includes an EPICS sequencer and database, defined during the RCG make process. As part of the sequencer, there is a section that reads in the data acquisition channel list file.

- awgtpman: This task sets testpoints and calculates and passes arbitrary waveform data to the real-time software.
- Daqd: This is the data acquisition software that stores and retrieves data to/from disk. In the standalone version, this has a built in NDS, which provides real-time and stored data retrieval services to any computer on the network.
- GDS daemons: There are several GDS daemons that provide network address information to the various DAQ/GDS tools. These require the setup of special files.

Real-time tasks may be run on core 1 and above, up to 16. These tasks are developed using the RCG. While in the diagram above, two code modules are shown (DAQ/GDS and Control), it is actually a single code thread. At runtime, this code is inserted to the kernel and locked to a specific CPU to run. In this fashion, this CPU core and real-time code are not under the control of the Linux scheduler. Rather, the task is slaved to the ADC inputs, which are in turn locked to the GPS clocks.

6.1.2 Distributed System Code Modules and Data Flow

In a distributed CDS, there are one or more FE computers and one or two DAQ computers (frame builders). The software architecture in this case is shown below. Primarily, the DAQ software is moved to the frame builder and a real-time network (FE DAQ NET) is used to communicate data from the FE computers to the frame builders. Another real-time network (FE PCI NET) communicates control data between FE machines.

In the distributed case, there is also a separate NDS process on the frame builders. Note that one, and only one, set of GDS daemons continues to operate in this system. While shown in a FE computer in the diagram, these daemons may run on any computer on the FE LAN, typically the FE boot server or FE file server.

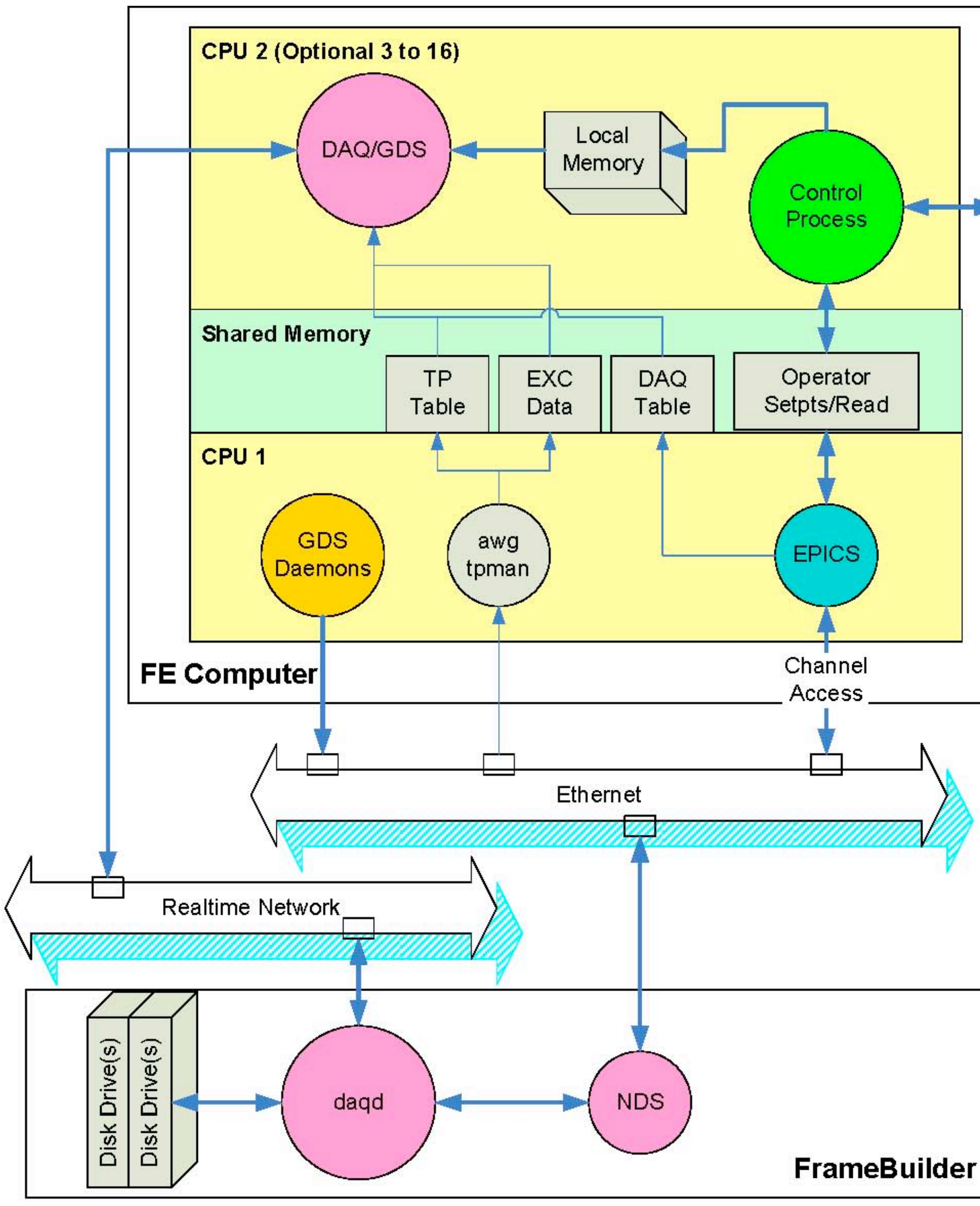

Figure 3: Distributed System Software Modules

Figure 3 Distributed System Software Modules

6.2 Software Installation

The source software for the real-time software development and generation are available via a checkout of the CDS aLIGO source tree from the LIGO CVS repository. EPICS components are available for download from the EPICS home page.

Various supporting software components, such as dataviewer and DTT, are in different CVS branches. At the moment, these components are typically downloaded as binaries, with this to change fairly soon to allow source downloads.

6.2.1 Common Directories

For aLIGO, the CDS systems use a common directory structure, as detailed in Reference C. This specifies where the real-time and application software is located. In overview, there are 5 top-level directories:

- /opt/cdscfg – Holds scripts to automate localization of paths, environment variables. This is detailed in Reference D
- /opt/rtcds – Holds real-time and DAQ executables, files and software build tools
- /opt/rtapps – Holds applications and libraries (EPICS, FrameCPP) for building real-time software
- /opt/cds – Holds control-room (operator) workstations tools, scripts and files
- /opt/apps – Holds applications and libraries (EPICS, MATLAB, GDS, Dataviewer) used on workstations

6.2.2 Localization by site and ifo

Reference D details how the directory tree under /opt/rtcds and /opt/cds is localized based on the site and ifo using files and scripts in /opt/cdscfg. These include the interferometers (sites cit, geo, lho, llo and ifos c1, g1, h1, h2, l1) but also localities for test stands (site tst and ifos x1, x2). All real-time software for a location should be under */opt/rtcds/<site>/<ifo>*.

Basic requirements are to create a file named after the site in /opt/cdscfg/site and for the ifo in /opt/cdscfg/ifo. Then copy a set of configuration scripts into /opt/rtcds/<site> and /opt/rtcds/<site>/<ifo>. For example, for site 'llo' and ifo 'l1':

```
cd /opt/rtcds/site
touch llo
cd /opt/cdscfg/ifo
touch l1
mkdir /opt/cdscfg/llo
cp /opt/cdscfg/tst/* /opt/cdscfg/llo
cp /opt/cdscfg/tst/x1/* /opt/cdscfg/llo/l1
```

- Note that current copies of these files are in the CDS Subversion under 'cds/trunk/cdscfg'

6.2.3 Real-time Build Software

The real-time models and build software are stored in two locations (See reference C). The RCG software, scripts and reference models are under */opt/rtcds/<site>/<ifo>/core*. At this location, the RCG software is checked out of the CDS Subversion repository 'advLigoRTS'. The top-level directory with the RTS makefiles is at */opt/rtcds/<site>/<ifo>/core/advLigoRTS/trunk*.

6.2.4 Real-time Executable Software Tree

The execution of 'make install-*mdl'* results in the executable code being moved into the following directories:

- EPICS interface software is copied to /opt/rtcds/<*site*>/<*ifo*>/target/<*ifo*><*mdl*>epics. For example an MIT build (<*site*> mit) for IFO 1 (<*ifo*> m1) with a model name of 'abc' would end up in */opt/rtcds/mit/m1/target/m1abcepics*.
- Using the same example, the real-time control software will be copied to */opt/rtcds/mit/m1/target/m1abc*.
- The GDS parameter files needed by awgtpman are copied into /opt/rtcds/<*site*>/<*ifo*>/target/gds/param, with a startup script produced in /opt/rtcds/<*site*>/<*ifo*>/target.
- Start and kill scripts are created in /opt/rtcds/<*site*>/<*ifo*>/scripts. For example, */opt/rtcds/mit/m1/scripts/startm1abc*. Note that:
    - This top level script actually invokes individual startup scripts in the EPICS, real-time code and gds (for awgtpman) target directories.

- Given the above, each component may be separately started by the their own startup command, with EPICS first. This can aid in the process of debug or allowing start/stop of new real-time code when the EPICS part does not change.

6.2.5 Operational Support Software

The limited operational support software to build real-time systems is under /opt/rtapps. The larger set of operational software for operator workstations is under /opt/apps. The sub-directory contents can currently be downloaded from the LLO CDS system at /data/advLigo/cds. A better distribution system is in development.

6.3 Environment Variables

As detailed in Reference D, standard setup scripts can define all the required environment variables, and should be included in the .bashrc and .cshrc account startup scripts. Also included are standard scripts to localize scripts in a variety of script languages (Bash shell, Perl, Tcl). Environment variables defined include 'site', 'ifo', PATH, LD_LIBRARY_PATH, etc.

6.4 Real-time System Power Up Conditions

Real-time FE systems need to have certain services set to execute on power up. This can be done in a number of ways, but the following examples show the use of the /etc/rc.local file to start the necessary services. A description of what each line does follows.

```
touch /var/lock/subsys/local
/opt/rtapps/epics/base/bin/linux-x86_64/caRepeater& // Starts Epics
channel access repeater
/opt/rtldk-2.2/rtlinuxpro/modules/rtcore&   // Starts RT linux core
sleep 10      // Wait for RT core, needed by remaining lines
/sbin/rm -f /rtl_mem_*    // Remove any left over shared memory
pointers
/etc/setup_shmem.rtl sam psl &   // Setup shared memory blocks
#ln -s /dev/rfm2g0 /dev/daqd-rfm   // Needed if using reflected memory
cards
#/sbin/service gm start    // Start myrinet network, if used
```

- Line 2 (caRepeater), starts the EPICS CA repeater task. This improves the efficiency of CA communications. This line needs to point to the installed caRepeater executable.
- Line 3 (rtcore) starts the real-time core tasks, required for operation of all real-time tasks to be loaded on this machine.
- Line 4 (sleep) allows time for rtcore task to fully install before running any other tasks.
- Line 5 (rm) removes any stale shared memory blocks
- Line 6 (setup) creates the shared memory blocks for real-time to non-real-time task communications. This line needs to be of the form
  /etc/shm_mem xxx yyy zzz &
  - where xxx, yyy, zzz, etc. are the three letter acronyms for real-time models which will run on this computer.
- Line 7 (rfm) is optional and not typically used. It is only for systems, such as in eLIGO, where systems must communicate via reflected memory networks.
- Line 8 (gm) starts up the myrinet network services. This is not required for standalone systems.

6.5 Diagnostic Daemons

One, and only one, computer on a network must be set up to run the GDS daemons. These daemons provide information to GDS tools running on the network as to where various resources may be found. These daemons are configured by providing service files in the */etc/xinetd.d* directory, which in turn provide locations of daemon executables and supporting files. The three GDS services that must be configured are described in the following subsections.

Once the daemons have been configured, the computer that is to run them should be rebooted. The status information may then be obtained by running *diag –i* from the command line of any computer on the network.

6.5.1 Channel Configuration

An example chnconf file is listed here, with configurable parameters listed below.

```
service chnconf
{
        disable         = no
        type            = RPC
        rpc_version     = 2-3
        socket_type     = stream
        protocol        = tcp
        wait            = yes
        user            = root
        server          = /opt/rtapps/gds/linux-x86_64/bin/chnconfd
        env             = HOME=/opt/rtcds/geo/g1/target/gds
        server_args     =
/opt/rtcds/geo/g1/target/gds/param/tpchn_G1.par /opt/rtcds/geo/g1/
target/gds/param/tpchn_G2.par
}
```

'user' must be set = root
'server' points to the location of the executable service file, typically set as above, with appropriate <site> and <ifo> specifiers, such as geo, cit, llo, tst, etc.
'env' points to the GDS home directory for finding GDS parameter files
'server_args' list the GDS parameter files which are to be read by the server. This list should contain all parameter files for all real-time processes on the network.

6.5.2 Diagnostics Configuration

This service file points to a second GDS daemon, with an example file listed below.

```
service diagconf
{
        disable          = no
        port             = 5355
        socket_type      = dgram
        protocol         = udp
        wait             = yes
        user             = root
        passenv          = PATH
        server           = /opt/rtapps/gds/linux-x86_64/bin/diagconfd
        env              = HOME=/opt/rtcds/geo/g1/target/gds
        server_args      =
/opt/rtcds/geo/g1/target/gds/param/diag_G.conf
}
```

All variable through passenv should be set as shown.
'server' points to the daemon executable.
'env' is set to the GDS home directory where parameter files exist
'server_args' points to the diagnostic configuration file

6.5.3 Leap Second Configuration

A GDS leap second daemon adds leap seconds to the GPS time at appropriate intervals, as listed in the leap.config file.

```
service leapconf
{
        disable          = no
        type             = RPC
        rpc_version      = 2-3
        socket_type      = stream
        protocol         = tcp
        wait             = yes
        user             = root
        server           = /opt/rtapps/gds/linux-x86_64/bin/leapconfd
        env              = HOME=/opt/rtcds/geo/g1/target/gds
        server_args      =
/opt/rtcds/gewo/g1/target/gds/param/leap.conf
}
```

Fields through 'user' should be set as above.
'server' points to the executable daemon file.
'server_args' points to the leap second file to apply.

6.6 Diagnostic Parameter Files

The diagnostic parameter files are located in /opt/rtcds/<site>/<ifo>/target/gds/param

6.6.1 Awg.par

```
[A1-awg0]
hostname=10.2.13.58

[A1-awg1]
hostname=10.2.13.58
prognum=0x31001004
```

6.6.2 diag.conf

The diag.conf file lists the available services and the IP addresses or names of computers that provide that service. An example is listed below.

```
# every line represents a configuration service
# the format is 'interface' 'ifo' '#' 'hostname' 'port/prog #'
'version'
# If a interface description starts with a '?', the service will only
# be announce if the corresponding machine answers to a ping.
# If an interface name starts with a '&', the host name will be looked
# up and replaced with its IP dot notation. Ping and host look up can
# be combined with '&?'.
# lines starting with # are ignored.
#

&nds * *  10.0.100.1 8088 *
&chn * *  10.0.100.2 822087685 1
&leap * * 10.0.100.2 822087686 1
&ntp * *  ntpServer * *
&err 0 *  10.0.100.2 5353 *
```

'nds' gives the name/address of the network data server. In a distributed system, this points to a frame builder.
'chn' points to the GDS channel server.
'leap' points to the GDS leap second server
'ntp' points to the network time protocol server.

‘err’ points to the computer which logs errors.

Typically chn, leap and err point to the same computer, namely the computer running the GDS daemons.

6.6.3 testpoint.par file

For the various tasks to find the appropriate awgtpman task to set and operate testpoints, a ‘testpoint.par’ files must be created which lists the network location for each instance. An example file is shown below.

```
# defines the rpc parameters of the gds test point controller
# test point nodes are numbered from 0 to 1; their section name is
"node#"
#   parameters are "hostname" (mandatory), "prognum" (default
0x31001001)
#   "progver" (default 1).

[A-node0]
hostname=10.1.100.23
system=psl

[A-node1]
hostname=10.1.100.23
system=sam
prognum=0x31001003
```

For each awgtpman task, this file must contain:

- A node name and number, in braces. The first letter is the site (G in this example for GEO), followed by ‘node’ and number. This number must match the GDS number assigned to the application model set in the user model file (see Reference A).
- Hostname = IP number or name of the computer which runs the associated awgtpman program.
- System = is set to the 3 letter name of the model associated with this awgtpman.
- Prognum = This is not required unless a single computer is running more than one real-time task created from a model file. In this example file, both the psl and sam tasks run on the same computer (separate cores), and therefore the second must have a prognum set. This number for a second awgtpman must be set to a value of 2 higher than the default. If a third is running on the same machine, then its number must be set to 2 higher than the second and so forth.

6.6.4 tpchan_XX.par

The ‘tpchan_xx.par’ files are automatically generated by the RCG and installed during the make install-daq-sys process. This file contains a list of all available excitation channels testpoints for that system. It should not be hand edited. A partial example is shown below.

```
#sam excitations
[G1:SAM-ETMX_LL_SEN_EXC]
ifoid = 1
rmid = 1
dcuid = 15
chnnum = 20001
datatype = 4
datarate = 2048

[G1:SAM-ETMX_LL_SEN_IN1]
ifoid = 1
rmid = 1
dcuid = 16
chnnum = 30001
```

```
datatype = 4
datarate = 2048

[G1:SAM-ETMX_LL_SEN_IN2]
ifoid = 1
rmid = 1
dcuid = 16
chnnum = 30002
datatype = 4
datarate = 2048

[G1:SAM-ETMX_LL_SEN_OUT]
ifoid = 1
rmid = 1
dcuid = 16
chnnum = 30003
datatype = 4
datarate = 2048
```

6.7 Frame builder

The installation of the frame builder software involves:

- Installing the daqd executable into the /opt/rtcds/<site>/<ifo>/target/fb directory.
    - Note that there are two versions, one for running on a standalone system and one to run on a separate frame builder computer in a distributed system configuration.
- Installing the nds software into the same target directory.
    - This is only required if running with the distributed system configuration. For standalone systems, daqd has this feature built in.
- Creating a master configuration file.
- Installing and modifying the daqdrc file.

Creating and modifying the configuration files is described in the following subsections.

6.7.1 master file

To determine which channels are continuously being transmitted by the various FE computers, including rates, data types, whether or not to save the data, and to get a list of all channels available as testpoints, the FB must read various configuration files. These are the .ini files, configured by the user, and .par files, automatically generated during the RCG build and install process. This file must be named 'master' and placed in the */opt/rtcds/<site>/<ifo>/target/fb* directory. An example file is listed below.

```
/opt/rtcds/geo/g1/chans/daq/G1SAM.ini
/opt/rtcds/geo/g1/chans/daq/G1PSL.ini
/opt/rtcds/geo/g1/target/gds/param/tpchn_G1.par
/opt/rtcds/geo/g1/target/gds/param/tpchn_G2.par
```

6.7.2 daqdrc File

The daqdrc file provides various user settable parameters for use by the FB software (daqd). An example file is listed below, with explanations as comments above each line.

```
**********************************************************************
# Uncomment following line only if running at Caltech 40m lab
#set cit_40m=1;

#At present, FB timing is controlled by one of the FE computers on the
network. The FB code needs to
#know the dcuid of this FE, along with its data rate.
set dcu_rate 10=32768;
set controller_dcu=10;

#Set FB to monitor dcu status. Typically set to 1.
set dcu_status_check=1;

#Following flag sets FB debug level. This is typically set to zero.
set debug=2;

#Following flag indicates to FB what to do with data which is out of
sync, where 0 is to continue to store values as they appear from FE, or
1, which indicates that FB should set all faulty data to zero.
set zero_bad_data=0;

#Point to master file
set master_config="/opt/rtcds/geo/g1/target/fb/master";

# Required line which starts FB configuration.,
configure channels begin end;

#Provide the name or address of computer which has GDS daemons running.
set gds_server = "geo" "geo" 0 0;

#??
set gps_leaps = 820108813;

# Set detector specific information which is to be stored as part of
the frame header.
set detector_name="CIT";
set detector_prefix="C2";
set detector_longitude=-90.7742403889;
set detector_latitude=30.5628943337;
set detector_elevation=.0;
set detector_azimuths=1.1,4.7123889804;
set detector_altitudes=1.0,2.0;
set detector_midpoints=2000.0, 2000.0;

enable frame_wiper;

#Set the total number of full frame directories.
set num_dirs = 10;
#Set the number of frame files per directory.
set frames_per_dir=225;
#Set the number of frames per file.
set full_frames_per_file=1;
#Set the number of seconds per frame file. This should be a number 2^n.
set full_frames_blocks_per_frame=16;
#Point to the base directory for full frame data, along with file
prefix and suffix.
set frame_dir="/frames/full/data", "M-R-", ".gwf";
scan frames;

enable trend_frame_wiper;
#Set the number of trend frame directories available.
set trend_num_dirs=10;
#Set the number of trend frame files per directory.
set trend_frames_per_dir=1440;
```

```
#Define the second trend base directory.
set trend_frame_dir= "/frames/trend/second/data", "M-T-", ".gwf";
 #Set the raw minute trend base directly
set raw-minute-trend-dir="/frames/trend/minute/raw";

set nds-jobs-dir="/opt/rtcds/geo/g1/target/fb/jobs";

enable minute-trend-frame-wiper;
#Set up minute trend directories.
set minute-trend-num-dirs=10;
set minute-trend-frames-per-dir=24;
set minute-trend-frame-dir="/frames/trend/minute/data", "M-M-", ".gwf";

scan minute-trend-frames;
scan trend-frames;
scan frames;

start main 30;
start profiler;

# comment out this block to stop saving data

#start frame-saver;
#sync frame-saver;
#start trender;
#start trend-frame-saver;
#sync trend-frame-saver;
#start minute-trend-frame-saver;
#sync minute-trend-frame-saver;
#start raw-minute-trend-saver;

#start frame-writer "225.225.225.1" broadcast="131.215.113.0" all;
#sleep 5;

start producer;
start epics dcu;
# Send out status via EPICS channels listed
start epics server "G0:DAQ-FB0_" "G1:DAQ-FB0_";

# Listener port for frame builder command line tool
start listener 8087;
# Listener port for Network Data Server
start listener 8088 1;
sleep 60;
clear crc;
************************************************************************
```