

LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Document Type LIGO-P990020-00 - E 04/09/1999
The LIGO Data Analysis System
S. Anderson, K. Blackburn, A. Lazzarini, W. Majid, T. Prince, R. Williams

Distribution of this document:

LIGO

This is an internal working document
of the LIGO Project.

California Institute of Technology
LIGO Project - MS 51-33
Pasadena CA 91125
Phone (818) 395-2129
Fax (818) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project - MS 20B-145
Cambridge, MA 01239
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

WWW: <http://www.ligo.caltech.edu/>

THE LIGO DATA ANALYSIS SYSTEM

S. ANDERSON, K. BLACKBURN, A. LAZZARINI, W. MAJID, T. PRINCE, R. WILLIAMS
*California Institute of Technology, LIGO Laboratory, MS 18-34,
Pasadena, California, 91125, USA
presented by Kent Blackburn
LIGO-P990020-00-E*



The Laser Interferometric Gravitational Wave Observatory (LIGO) will achieve astrophysically interesting strain sensitivities beginning in the year 2000. In order to extract the astrophysical signals from the data collected, LIGO is developing the LIGO Data Analysis System (LDAS) which will provide on-site and off-site analysis of the data sets collected as part of normal LIGO operations. This LDAS computational analysis system will be discussed.

1 Introduction

The Laser Interferometer Gravitational Wave Observatory (LIGO)¹ will search for direct evidence of gravitational waves emitted by astrophysical sources in accord with Einstein's General Theory of Relativity. The indirect evidence of gravitational waves has already been demonstrated for the binary neutron star system PSR1913+16^{2,3,4}. Detection of gravitational waves by LIGO from the late stages of such compact binary systems will require that optimal filtering algorithms match the output data from LIGO against greater than 10^4 to 10^5 waveform templates, each containing of the order 10^5 data points⁵.

The initial configuration for LIGO will include three interferometers located at two sites separated by 3000 kilometers and synchronized by GPS time stamps. These observatories will collect data from over 1000 channels at an aggregate data rate estimated to approach 10 megabytes per second. Selecting out the reduced data needed to carry out the LIGO data analysis from such data is a demanding communications task for the interface into the full data sets. To facilitate this task LIGO will use highly structured data formats and develop a layered software architecture allowing high level communications using the TCL interpreter based command language, along with an optimized data manipulation layer based on ANSI C and C++ compiled code

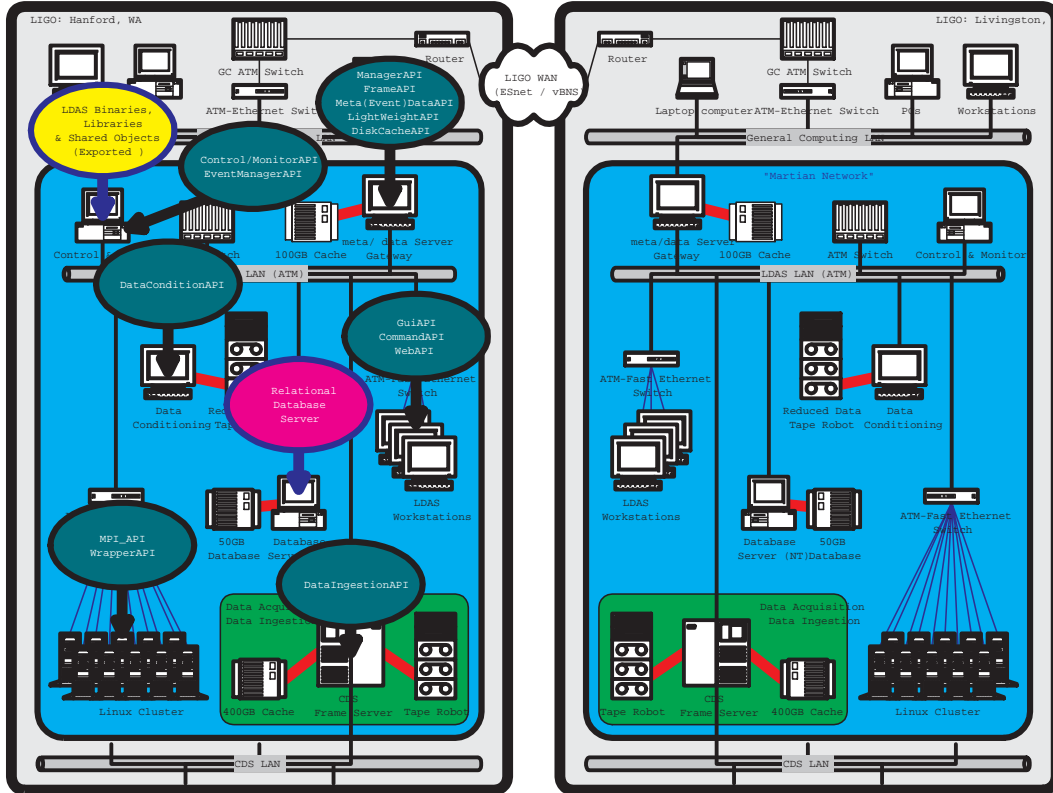


Figure 1: LIGO's on-site LDAS functional hardware diagram.

modules. Such a configuration will allow LIGO to distinguish signatures in the data stream of astrophysical origins from terrestrial events, perform coincidence observations between the two sites (and to a lesser extent, among three interferometers), correlate multi-detector outputs, and regress coherent noise sources from the gravitational wave channel.

The computational analysis system that LIGO is developing to support the initial configuration of LIGO consists of self-similar architectures being placed both at the two sites (on-site) and at the data archive center (off-site). This architecture support user interfaces, data selection, signal reconstruction, parallel computation, database management, data caching and data archiving. Figure 1 illustrates the on-site architecture while Figure 2 illustrates the off-site architecture.

The LIGO Data Analysis System (LDAS) system will be equipped with the signal analysis algorithms needed by LIGO to search for gravitational waves from the inspiral of compact binary systems composed of neutron stars and black holes, the ringing of the event horizon of a black hole, the periodic oscillations of a rotating neutron star with a quadrupole mass distribution, a supernovae strain burst seen in coincidence with other detectors, a stochastic gravitational wave background, and sources which we haven't yet imagined using correlation techniques and time-frequency signal analysis methods⁶.

2 Production Data Flow

The flow of data within the LDAS is dynamic and tunable to allow the most scientific gain from the data collected by LIGO. The system will however, maintain a production flow of data which directs data from the instruments into the parallel filter banks where many thousands of template waveforms will be matched against the content of LIGO strain data. This production

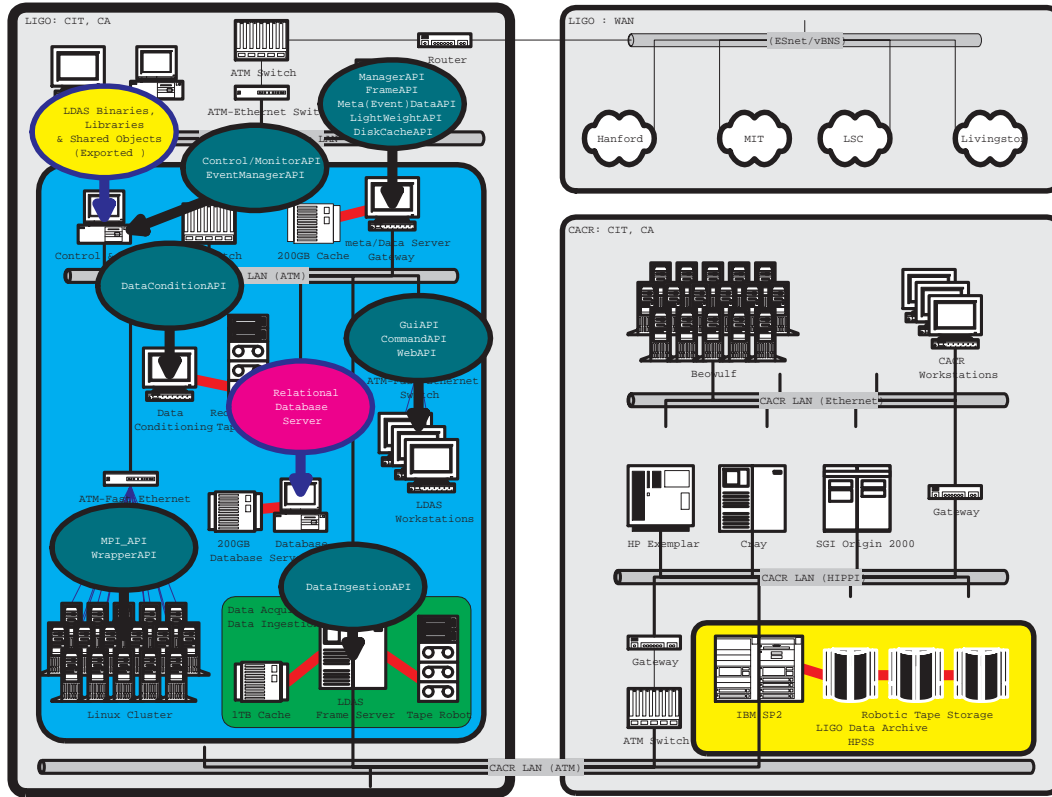


Figure 2: LIGO's off-site LDAS functional hardware diagram.

data flow is outlined here.

Data from the LIGO interferometers is sampled and digitized by the Control and Data System (CDS). These digitized signals are collected at rates as high as 16384 samples per second from each of hundreds of signal points within the interferometers (IFO) and the Physical Environmental Monitoring systems (PEM). Each second the digitized signals are collected by the Frame Builder within the Data Acquisition System (DAQS) and written to file in the International Gravitational Wave Detector Format⁷ (IGWD). The Frame Builder records these one second frames to a large disk cache capable of storing half a days data from each interferometer. These Frames are then placed on magnetic tape for long-term storage and later shipping to the archive center. As new Frame files appear on the disk cache, they are recognized by LDAS which opens each new Frame, using the **frameAPI**, and writes summary statistics and metadata about the Frame into the LDAS database and passes the contents along the pipeline. Also during this epoch, the parallel task of monitoring IFO and PEM health and status is being carried out by the Global Diagnostics System (GDS). The GDS filters generate triggers which detect non-astrophysical events that may contaminate the gravitational wave channel or indicate that an IFO is outside of its nominal operations. These triggers are ingested into a LDAS meta-database and will be used to make various cuts on the analysis cycle and data products.

Data selected out of the Frames by the **frameAPI** are then passed to distributed computing processes within LDAS (or remotely if needed) using a C++ socket communications library which provides distributed objects from an Internal LDAS Light Weight Data Format (ILWD) C++ class. These classes support collections of simple arrays for standard data types found in the Frames. It also provides a robust set of attributes used to specify the data within the objects.

The standard data flow will send selected data from the **frameAPI** to the **dataCondi-**

Binary Inspirational Template Compute Requirements

(estimated per interferometer with 8x overlap)

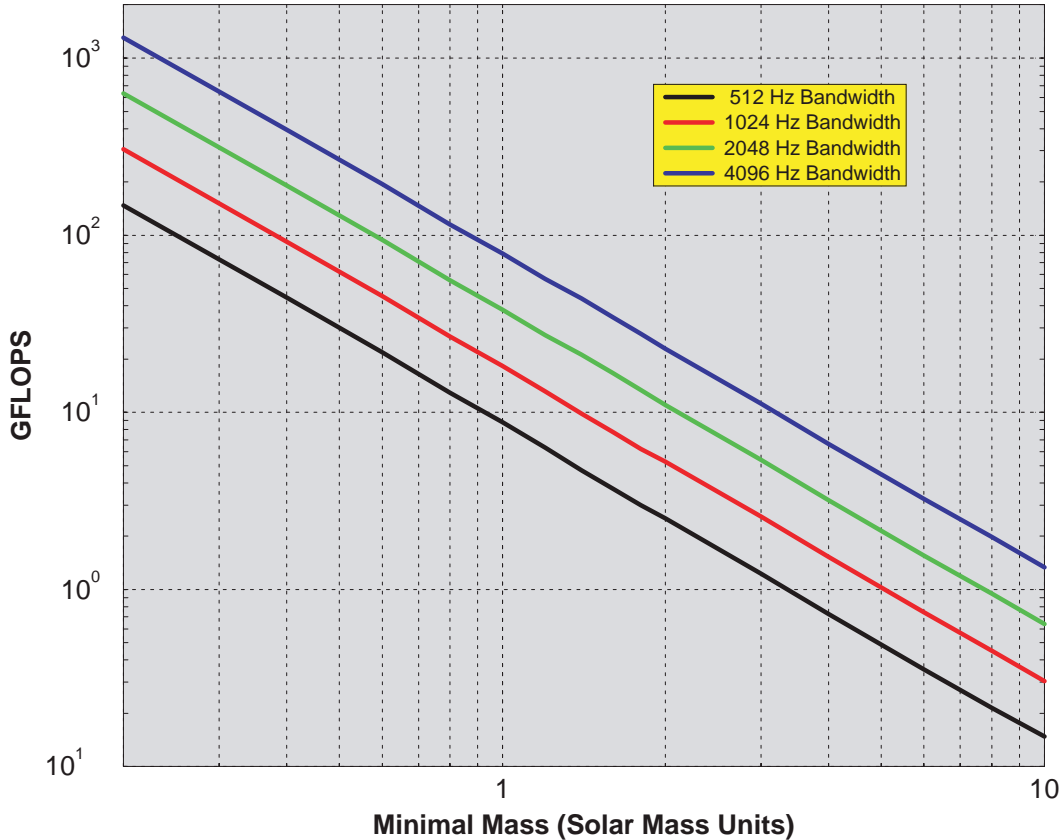


Figure 3: Compute performance required per interferometer signal to carry out a flat search for compact binary inspirals using the initial LIGO noise floor as a function of the smaller mass object in the binary system.

conditioningAPI which prepares the data for further analyses. This includes calibration, bandpass filtering, decimation, Fourier transforms, power spectrum estimation, Kalman filtering, cross-talk signal regression and other advanced digital signal processing of the data. The **data-ConditioningAPI** sends its data products throughout the LDAS pipeline to other distributed processes using the ILWD classes. These data products are used to add entries into the LDAS database and to carry out the search for gravitational wave signals in parallel computing components of LDAS.

Conditioned data is next distributed to the **mpiAPI** in the form of ILWD classes. The **mpiAPI** acts as the master process for all parallel computing processes within LDAS. It distributes the conditioned data to all the slave processes using the Message Passing Interface library standard (MPI). Each slave process applies its subset of the template bank of waveforms associated with the astrophysical searches being performed to the incoming conditioned data. The computational performance needed by LDAS to carry out astrophysical searches is assumed to be dominated by the need to carry out a flat search for inspiraling compact binaries. The amount of computation depends on the mass of the smaller object within the binary system and on the frequency support needed by the waveforms. Figure 3 illustrates this dependency for waveforms composed of frequency support of 512 Hz, 1024 Hz, 2048 Hz and 4096 Hz.

For this flat search on compact binary inspirals, using one solar mass as the minimum mass for the smaller companion and frequency support of 2048 Hz for the template waveforms, each LIGO interferometer would require roughly 40 gigaflops of computer performance to keep up with

data continuously piped out of the system. Such computer performance is easily achieved using a distributed cluster of fast PCs running Linux and MPI. All other searches for gravitational waves are expected to be of order 10% of this scale, with the exception of the all-sky pulsar search. If the flat search for compact binary inspirals is replaced with a hierarchical search as is expected by the time data analysis begins, then this demand could be significantly reduced.

The outputs of the astrophysical filters are collected by the **mpiAPI** master process where any further statistical distribution results are produced from the collective outputs. A final summary report classifying candidate events is then placed into the ILWD format and sent to the **eventManagerAPI**. The **eventManagerAPI** can then gather information from the LDAS database on possible corresponding diagnostic triggers that overlap with any candidate astrophysical events. If a refined search is needed on a tighter parameter space using higher order template waveforms, the **eventManagerAPI** will request a new MPI process via the **mpiAPI**. The **eventManagerAPI** will also update the astrophysical event tables in the LDAS database with any candidate events. This cycle then repeats using the next segment of data stored on disk in the Frame format.

3 LDAS Software Design

Many of the software components of the LDAS were mentioned in the discussion of the production data flow model. The Software design consists of modular blocks, each identifying a bounded functional purpose. The set of all LDAS software blocks is illustrated in Figure 4. In addition to the modules previously mentioned, there are **userAPIs** which provide graphical, Web, and command-line interfaces into the LDAS system. There is the **managerAPI** which is responsible for scheduling tasks. Parallel processing resources are configured and managed by the **controlMonitorAPI**. Disk cache and data ingestion functions are managed by the **diskCacheAPI** and the **dataIngestionAPI**. The LIGO Light-Weight Data Format, based on the eXtensible Markup Language (XML), is supported by the **lightWeightAPI** and provides users with a simple, easy to read and write data format which will soon be supported directly by web browsers. The system also supports remote processing outside of the scope of LDAS through the **remoteFilterAPI** module. There are also functional libraries used by the system to carry out data I/O using the Frame, LIGO Light-Weight and commercial database table formats. The system contains all of its numerical algorithms used for filtering and analysis within the filter library.

Each API is a stand-alone distributed processing environment based on TCP/IP socket communications. The software architecture of the LDAS API is layered into two main environments (See Figure 5), a command interface layer using the interpreted command language TCL/TK, with an embedded C++ layer used to extend the TCL/TK functionality with high performance C and C++.

The TCL/TK layer contains within its custom TCL/TK module, code which is unique to each LDAS API. It also contains a generic TCL/TK module which is common to all LDAS APIs and is used to standardize the methods for sending and receiving commands and messages between the many LDAS APIs. It also provides TCL/TK procedures for logging and providing online help.

The TCL/TK layer reads and evaluates a TCL resource file upon start-up. This resource file is used to configure the LDAS API and provides information about the environment in which the API is running. Examples of the types of information that can be registered in the resource file are the IP addresses of other important components in the LDAS system, the port numbers to assign server sockets, etc. The resource file is itself TCL code.

The C/C++ layer contains within its custom C/C++ module, code which is unique to each LDAS API. This code may manage SQL calls to the database or perform FIR or IIR filtering on

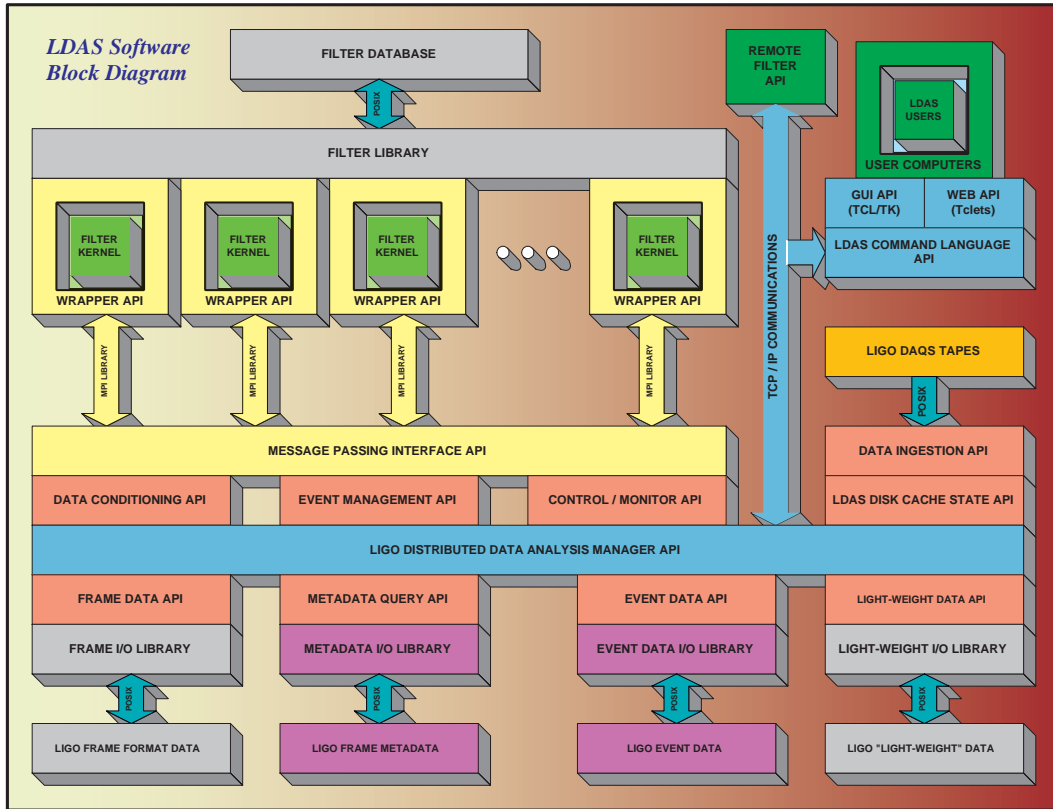


Figure 4: The LDAS Software Block Diagram.

discrete time series data sets, etc. It also contains a generic C/C++ module which is common to all LDAS APIs. Its primary functions are to manage the Internal LDAS Light Weight Data (ILWD) classes, communicate data represented by these objects between LDAS APIs using the C++ socket classes, and store and restore the state of objects contained within each LDAS API.

Each LDAS API manages three distinct classes of communication sockets. Two are within the TCL/TK layer and are referred to as the *operator* and the *emergency* communication sockets. The *operator* socket is the default socket type used to send or receive normal commands and messages. The *emergency* socket is used during exceptions to communicate abnormal conditions back to the **managerAPI**. Each of these communication sockets is isolated from the other by unique TCL interpreters and unique port assignments. The third class of communications socket is the *data* socket found in the C++ layer where it is used to send and receive binary data in the ILWD format between LDAS APIs.

This software design allows the LDAS to be distributed, isolating communication intensive components to the LDAS API's TCL/TK layers, while providing for efficient usage of CPU within the compiled C/C++ layer. The use of MPI provides a highly portable standard for carrying out parallel computing within the LDAS design. TCL/TK, C and C++ are highly standard, portable languages which are available on almost all popular computer platforms. The LDAS system will be developed on top of UNIX, using the POSIX operating system interface standard for increased portability at the level of various flavors of UNIX. Using TCL/TK as the *steering* language for computationally intensive processing allows greater flexibility in the configuration of LDAS at runtime, eliminating the need to recompile to alter data flow behavior. The use of TCL/TK as the *steering* language has other advantages. The graphical widgets associated with TK are now coupled with the releases of the TCL command language. There is also a TCL/TK plug-in for popular web browsers, allowing the same code to be run from web

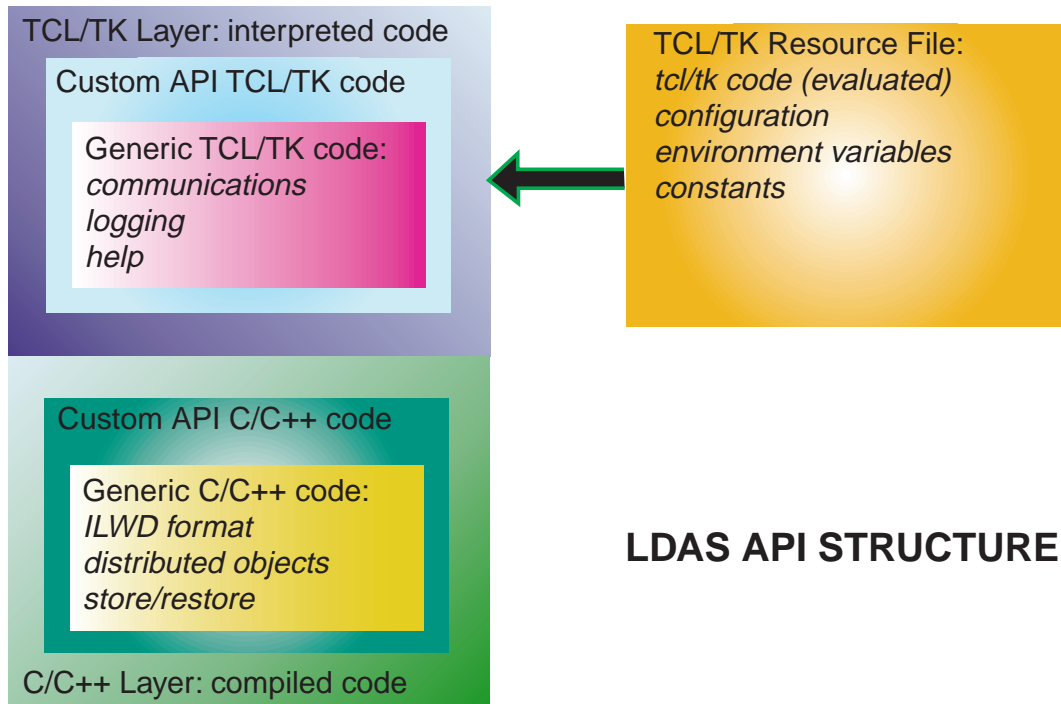


Figure 5: The LDAS API Layer Structure.

browsers as *TCLets*.

4 User Interfaces

Users interact with the LDAS system using **userAPIs** based on TCL/TK. These **userAPIs** provide command line, Graphical User Interfaces (GUI), and Web interfacing. They communicate with LDAS by sending messages requesting commands be performed by the system directly to the *operator* socket on the **managerAPI**. This is illustrated by Figure 6.

The **managerAPI** delegates a subprocess called the *assistant manager* to handle the specifics of the request. Multiple *assistant managers* can be started and simultaneously active at any given moment. These *assistant managers* carry out the appropriate sequence of commands within the framework of the LDAS to accomplish the user requested task, each independently sharing the system resources to schedule and sequence the user requests stored on the command queue.

The **managerAPIs** *operator* communication socket is always listening for requests to carry out a task. To protect this socket from break-ins from the internet, an authorization policy based on an access key is used. The **userAPIs** are required to communicate the key with each request. Each of the **userAPIs** shares a common high-level command language which supports variation through parameter arguments. It is also possible to send full length scripts to the **managerAPI** using the TCL language with the LDAS extensions.

The **userAPI** also benefits from the TCL socket command by having the ability to learn new behavior or update built-in procedures simple by uploading new TCL/TK scripts through the socket. This allows the **managerAPI** to customize the **userAPI's** presentation and functionality on the basis of the types of requests users place on the LDAS system. Figure 7 illustrates one such prototype **userAPI** which requests data from a data server and uploads the procedures to plot the data in an x-y graph, view remote file systems, interpret PERL code, and display HTML help files relevant to the service being provided to the client **userAPI**. Such flexibility

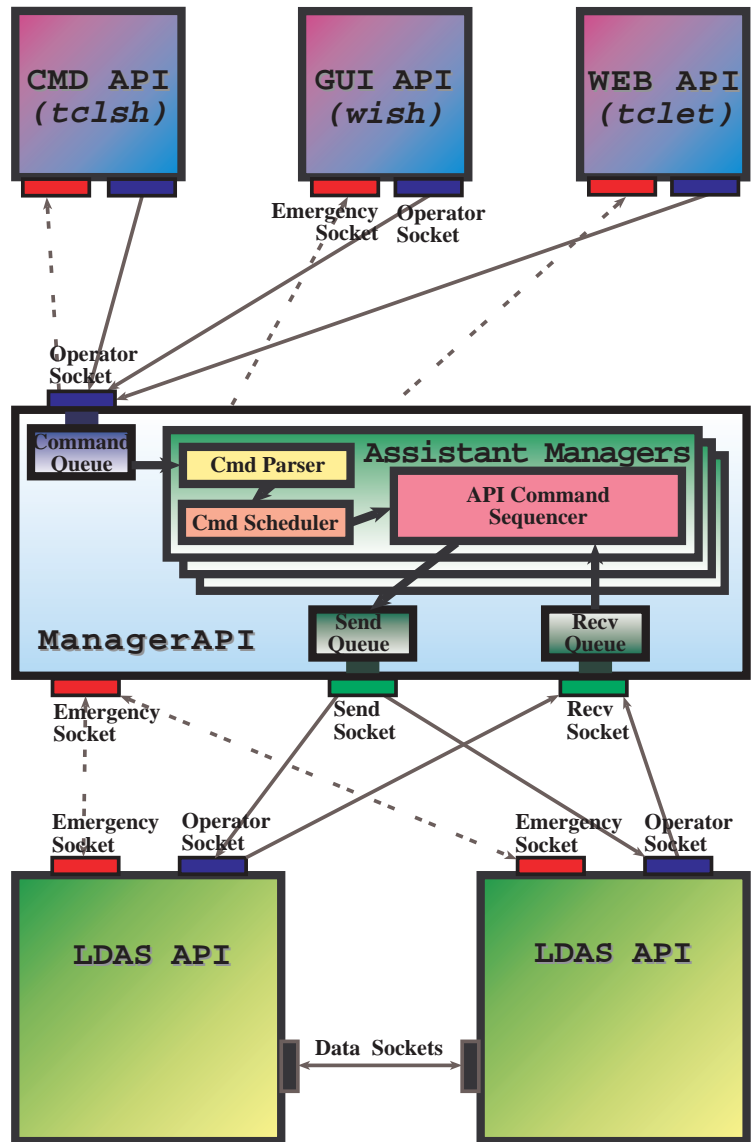


Figure 6: Users interface with the LDAS through **userAPIs** which connect to the **managerAPI**'s *operator* socket.

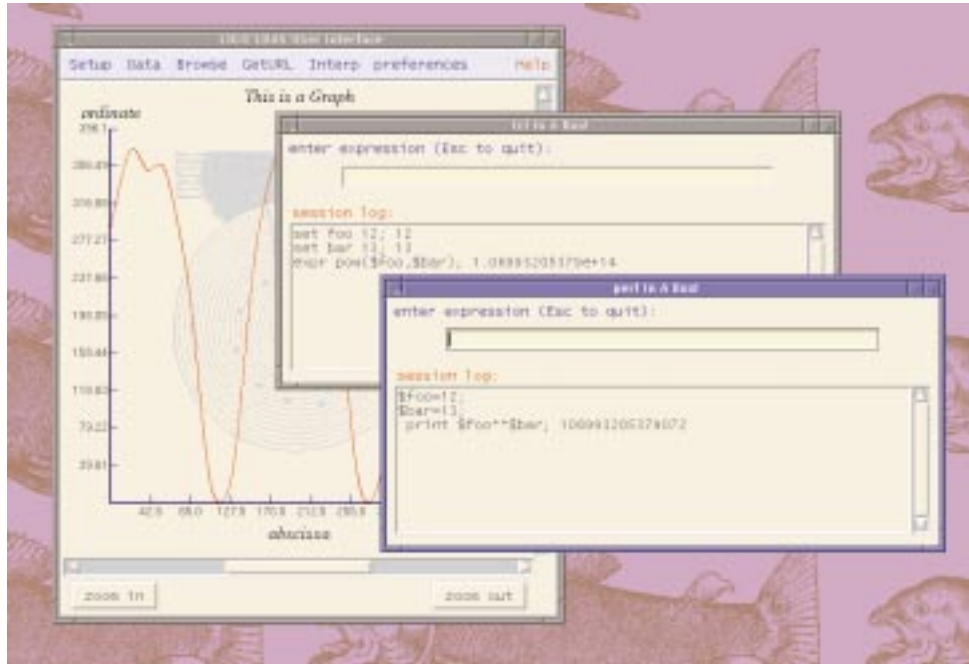


Figure 7: Users interface which *learns* functionality from the server on the basis of the requested task.

can easily be extended to TCL based **userAPIs** that are uploaded into web browsers by adding the appropriate security policy to the TCllet (in both the TCL code and in the browser's configuration files). All that is needed is the designate policy file which can easily be distributed by a password protected web page.

5 LDAS Hardware Design

The LDAS software runs over the UNIX operating system on hardware that supports POSIX and TCP/IP sockets. This allows for a distributed system that can be reconfigured as usage models change or evolve. In order to protect the LDAS from internet packets, the hardware will be placed on a *martian network* which does not register (or participate) with the larger internet. A gateway is used to connect the secure *martian network* with the outside world. The gateway (shown in Figure 1 and Figure 2) is used to present the **managerAPI**, the **frameAPI** and the **metaDataAPI** to the internet base of LDAS users. Using just these three APIs, all of LDAS is available for distributing data, metadata and to carry out user requested analysis tasks. However, only the **managerAPI** receives command inputs from the internet. The **frameAPI** and **metaDataAPI** will only transmit data products to the outside.

The astrophysical waveform template analysis is numerically dominated by Fourier transforms of approximately 10^4 waveforms on the order of 10^4 points each. However, this is an extremely parallel analysis task. For this reason, a cluster of PCs running Linux on a fast ethernet switching fabric will be more than adequate to perform the astrophysical analyses (excluding

the all-sky periodic source search which requires many orders of magnitude more compute performance to reach deep into the galaxy for all observations⁸). Such supercomputing topologies are often referred to as *BEOWULF* systems. LDAS will target three such systems, two at the sites and one at the archive composed of roughly 30 to 60 CPUs each.

LIGO will generate roughly 10 megabytes of data each second from its three interferometers. Of this data, the gravitational wave signals will make up only about 100 kilobytes each second. All of the data will be stored on the sites in spinning hard disk media for roughly 12 hours in a FIFO cache managed by the **diskCacheAPI**. Some subset of the data will also be archived in a small on-site tape archive capable of storing approximately 50 terabytes of data. The data will be copied to transport tapes used to deliver LIGO data to the LDAS central archive which will be sized to hold at least two years of the science data (expected to be on the order of 10% of the original raw data) along with the gravitational wave signal for all times. This central archive will be roughly 200 terabytes in size and will use IBM's High Performance Storage System (HPSS) running on IBM's SP2 nodes to maintain the archive. It is important to note that the computer industry is a rapidly evolving technology. These hardware plans may be overshadowed by newer technologies in the period between this writing and the time LIGO will make its major hardware purchases. These are the baseline solutions that exist today that meet LDAS requirements.

6 Status Report

The LDAS software is currently being implemented as outlined in this review. At the time of this writing roughly 20% of the software modules have been developed. An alpha release of a subsystem of LDAS software will be delivered on site to Hanford by Summer of 1999.

The LDAS hardware procurement is being staged in such a way as to deliver the most computer technology for the cost, while at the same time providing platforms for prototyping and development. LDAS has begun the purchasing of infrastructure for the sites. The central LDAS ATM switch has been delivered to Hanford, Washington). In addition, the orders are now being placed for the database and data distribution servers needed to provide data at Hanford during the installation of the first interferometer.

More information about the LIGO Data Analysis System can be found on the world wide web at the URL, http://www.ligo.caltech.edu/~prince/LDCG_lsc/LDCG.html. There you will find links to technical documents, presentations, and source code.

Acknowledgments

The LIGO Project is supported by the National Science Foundation under the cooperative agreement PHY-9210038.

References

1. A. Abramovici *et al*, *Science* **256**, 325 (1992).
2. R.A. Hulse & J.H. Taylor, *Astrophys J.* **191**, L59 (1974).
3. R.A. Hulse & J.H. Taylor, *Astrophys J.* **195**, L51 (1975).
4. R.A. Hulse & J.H. Taylor, *Astrophys J.* **191**, L55 (1975).
5. B.J. Owen, *Phys. Rev. D* **53**, 6749 (1996).
6. K.S. Thorne, *Gravitational Radiation*, in *300 Years of Gravitation*, eds. S.W. Hawking & W. Israel (Cambridge U. Press, Cambridge, UK, 1987).
7. LIGO & VIRGO, "Specification of a Common Data Frame Format for Interferometric Gravitational Wave Detectors", LIGO-T970130-B, VIR-MAN-LAP-5400-103.
8. P.R. Brady *et al*, *Phys. Rev. D* **57**, 2101 (1998).