LIGO Data Analysis System

Design Requirements Review (DRR)

0830 - 1200 PST **12 December 1997**

Agenda

- Requirements
- Conceptual Design
  - Hardware implementation
  - Software implementation
  - Networks
- Prototyping activities
- Schedule

LIGO Data Analysis System
Design Requirements Review

Comment to LIGO (A.L.) from an anonymous FNAL staff physicist (Assoc. Director for Scientific Technology and Laboratory Information):

"...Thank your lucky stars you don't have to deal with two competing collaborations, each with about 35 institutions and over 400 physicists, all of whom believe they are expert on computing, and which have no history of having their kimonos ripped open to external review...."

LIGO Data Analysis System (LDAS)
Requirements

- Assumptions/Dependencies
  - Detector delivers a fully functional DAQS. Data are written in frame format to a disc cache system available to the on-line LDAS. Iincludes the availability from DAQS of data-valid logic flags to identify saturated or aliased waveforms.
  - Detector implements interferometer diagnostics system. LDAS does not need to provide real-time (signal) feedback information to the LIGO interferometers. LDAS - Diagnostics System interface shall be primarily through the operator or scientist. Data or parameters derived by diagnostics routines will be done through frame-based data.
  - LDAS, together with DAQS, will provide for an on-line (volatile) data storage system capable of accommodating a volume of data sufficient to provide overlap between shifts.
  - LDAS goal shall be to process datastream at real-time rates and on-line. This includes providing for the exchange of detection event lists between LIGO sites.
  - The off-line system does not directly interface to the on-line system.
  - Data reduction shall be accomplished as far upstream in the data acquistion process as possible in order to enable LIGO to archive reduced datasets for at least 5 years. As a target, a minimum volume reduction of 10X is assumed. As a minimum, the GW channel, calibrated in strain, shall be archived permanently.

- Assumptions/Dependencies
  - Specifically not considered to be within the scope of the LDAS are:
    - Data analysis functions performed at centers other than the LIGO Laboratory Facilities.
    - The on-line diagnostics system used for stimulus-response characterization, transfer function determination, and calibration functions. However, it is expected that software developed for the LDAS will find utility within the diagnostics system.
    - Simulations shall be provided separately from, but coordinated with, the LDAS. The interface shall be using frame-based representations of simulation outputs.

LIGO Data Analysis System (LDAS)
Requirements

- Mission-critical services:

1. Provide on-line analysis at the observatories.

- » Physical strain extraction possibly using relevant ancillary channels (e.g., PEM) to remove instrumental or environmental signatures.
- » Processing of strain data through real-time detection algorithms for both performance monitoring and scientific purposes.
- » A means to cross-correlate data (either time series or event lists) from multiple interferometers.
- » A means to store data frames and analysis results (local to the Observatory LAN) to short term storage media. This functionality will be provided by the LIGO DAQS resources, with augmentation by LDAS.
- » A means to access both “live” and short term archived data via the Observatory LAN and the LIGO WAN. Access shall be subject to available bandwidth and demand.
- » Means to retrieve, concatenate and extract specific channels of recent data from the on-line storage system. A means to display and visualize results of analyses over the Observatory LAN
- » Sufficient automation to run continuously and autonomously during periods of normal operation.

LIGO Data Analysis System (LDAS)
Requirements (cont.)

- Mission-critical services:

2. Provide for extended off-line processing capabilities:

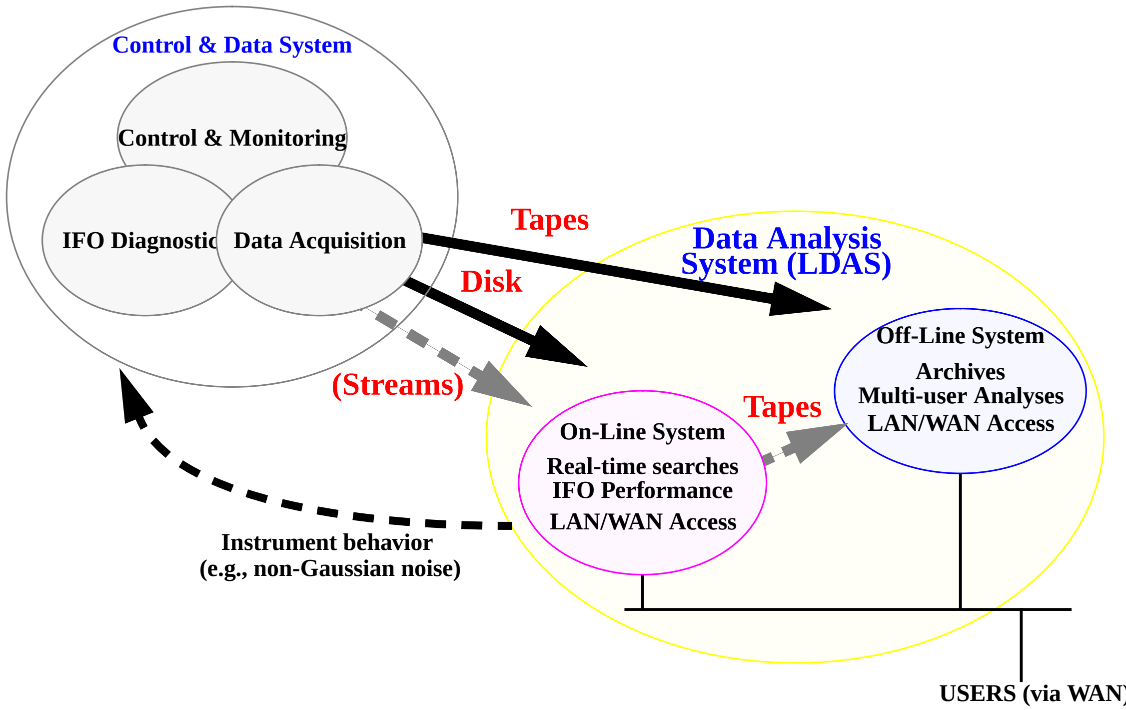
  - A means to reduce the raw data to science data representing calibrated GW strain data and a reduced subset of ancillary data and a data quality descriptor.
  - A means to archive, retrieve and distribute reduced datasets acquired over a period of time at least 5 years in duration.
  - A means for duplicating reduced datasets either for backup or for distribution.
  - Sufficient computing margin to enable multiple analyses to be conducted in parallel.

3. Provide a means to access the data archive via the LIGO WAN by the LIGO Laboratory and LIGO Scientific Collaboration to support database manipulation at the off-line site by remote users.

4. Implementation goals:

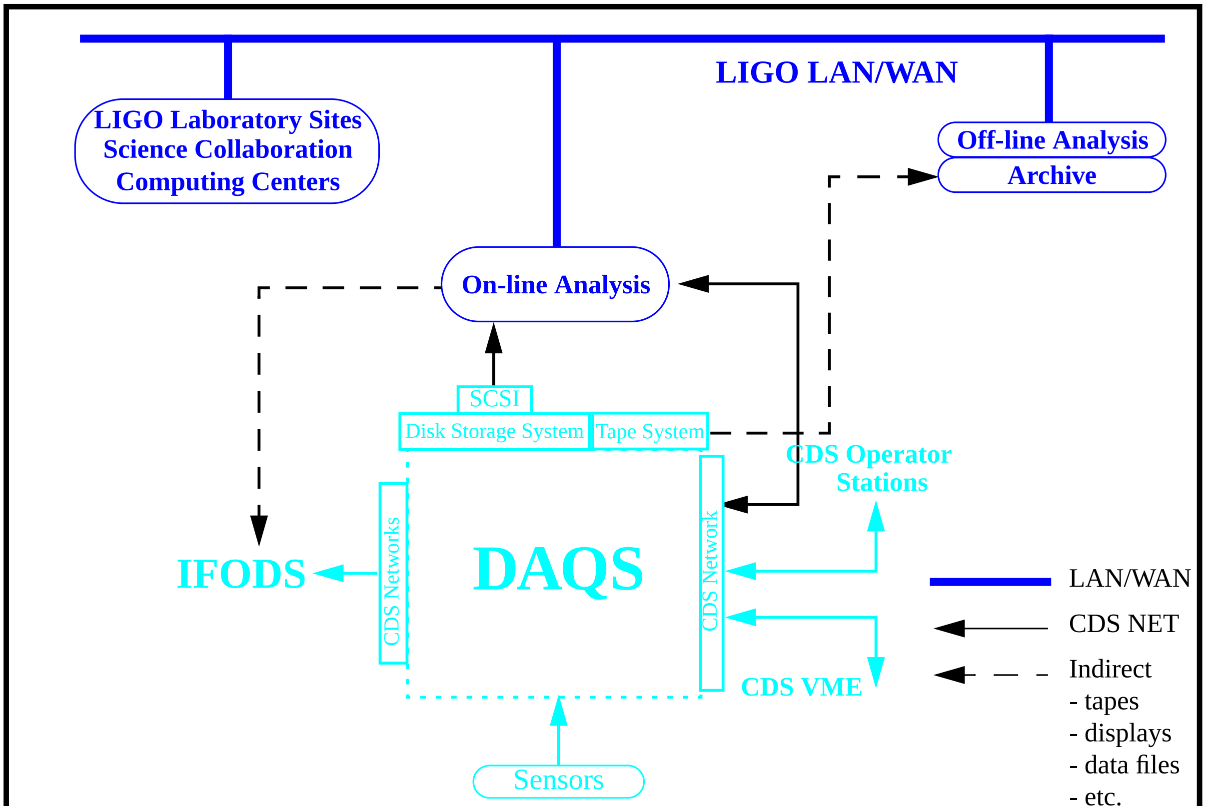
- Flexibility => No (or very little) custom hardware with custom software interfaces
- Extensibility => Modular (not function specific) component design
- Portability => Upgradable hardware under same software or vice-versa => POSIX compliance, software standards, etc.
- Maintainability => Object oriented programming design (“reusable software components”)

LIGO Data Analysis System (LDAS)
Functional Units



LDAS Functional Units. LAN or WAN refers to the local area networks either within a Laboratory site or between sites.

LIGO Data Analysis System (LDAS)
Interfaces



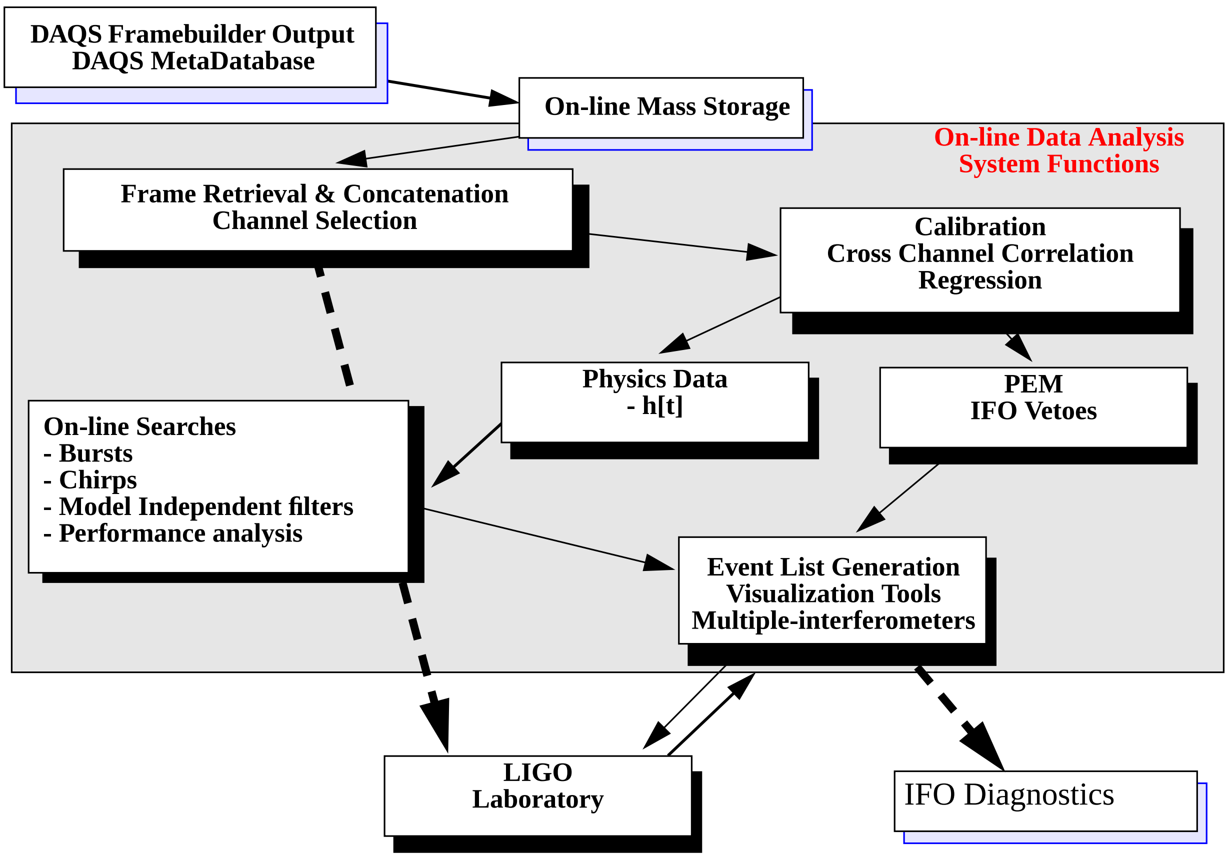
LDAS Interfaces

Hardware Design & Implementation

LIGO Data Analysis System (LDAS)
Design

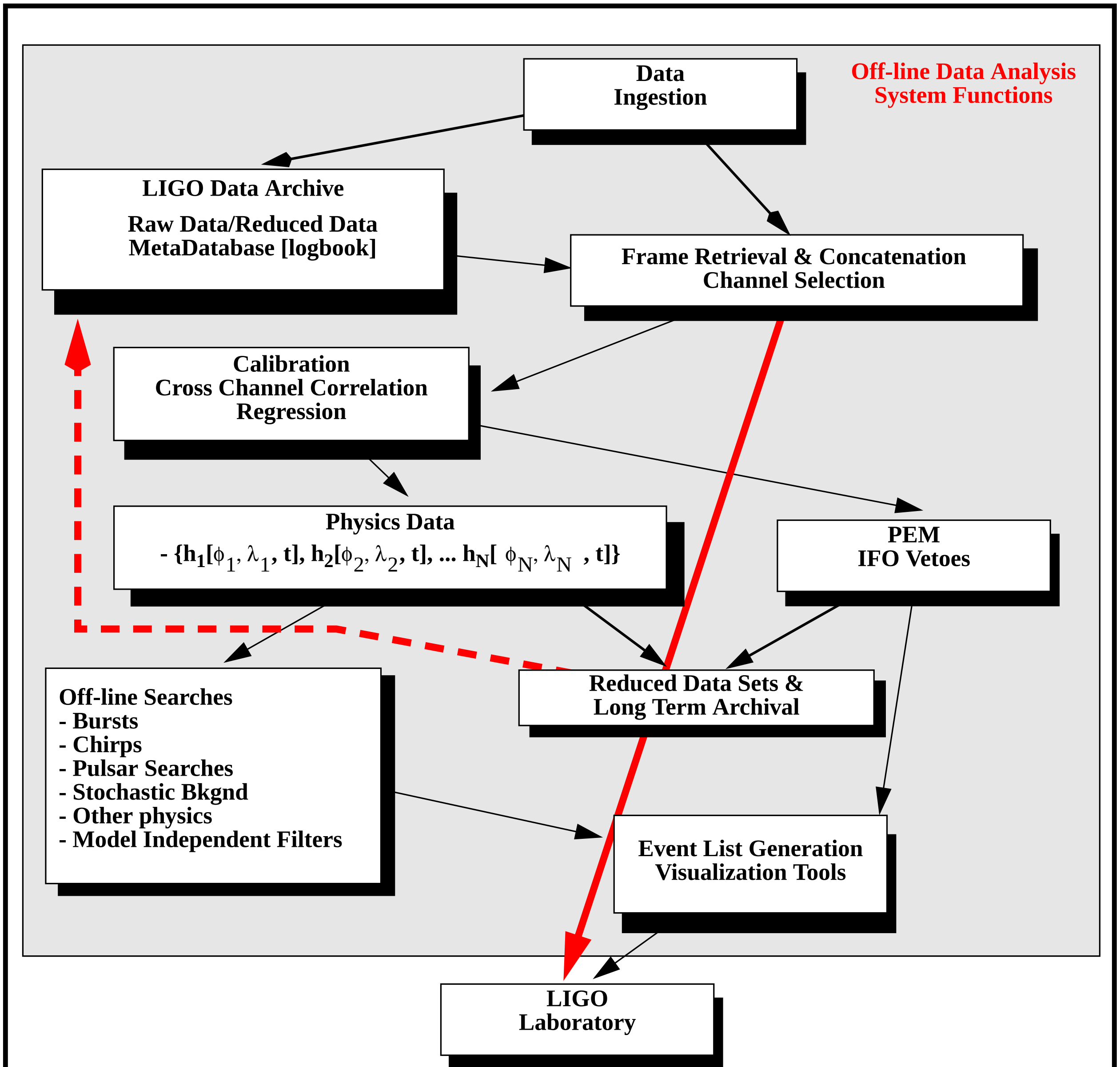
- Two LDAS components
  - On-line LDAS
    - Two systems, one for Hanford, and one for Livingston
    - Hanford system handles 2 interferometers
    - Provide computational power at the observatories to support diagnostics, detection, expansion/growth,...
  - Off-line LDAS
    - Collaborative arrangement with CACR
      - Dedicated LIGO hardware within CACR on scale of observatory systems
      - Database archive
      - Strategic use of other CACR facilities as available
    - Transparent access for off-line analysis of archived data
      - LIGO Laboratory
      - LIGO Scientific Collaboration
- Wide area network (WAN) to enable inter-site communications
  - University scientific and engineering support to Observatories
  - Access to archive database
  - Access to on-line data from observatories
  - Inter-observatory event sharing

LIGO Data Analysis System (LDAS)
On-line Functions



LDAS Flow Block Diagram: On-line Analysis

LIGO Data Analysis System (LDAS)
Off-Line Functions



LDAS Flow Block Diagram: Off-line Analysis

LIGO Data Analysis System (LDAS)
Functions

- On-line component
  - » diagnostics support
    - – tests not requiring closed loop feedback to interferometers
    - – data cross-correlation and regression
  - » characterization of sensitivity
    - – long term drifts -- normalization
    - – changes in spectral shape of noise floor
    - – non-Gaussian bursts -- instantaneous and time-averaged degradations
  - » signal processing for detection
    - – best-estimate strain extraction from signal regression
    - – processing of strain data in real-time detection algorithms employing various matched filtering techniques:
      - – inspiraling binary coalescences
      - – transients
        - supernovae
        - ringdowns
        - non-Gaussian event dictionary
      - – periodic sources
        - directed
        - limited parameter space search
      - – model-independent searches -- wavelets, etc.
      - – serendipitous events

LIGO Data Analysis System (LDAS)
Functions

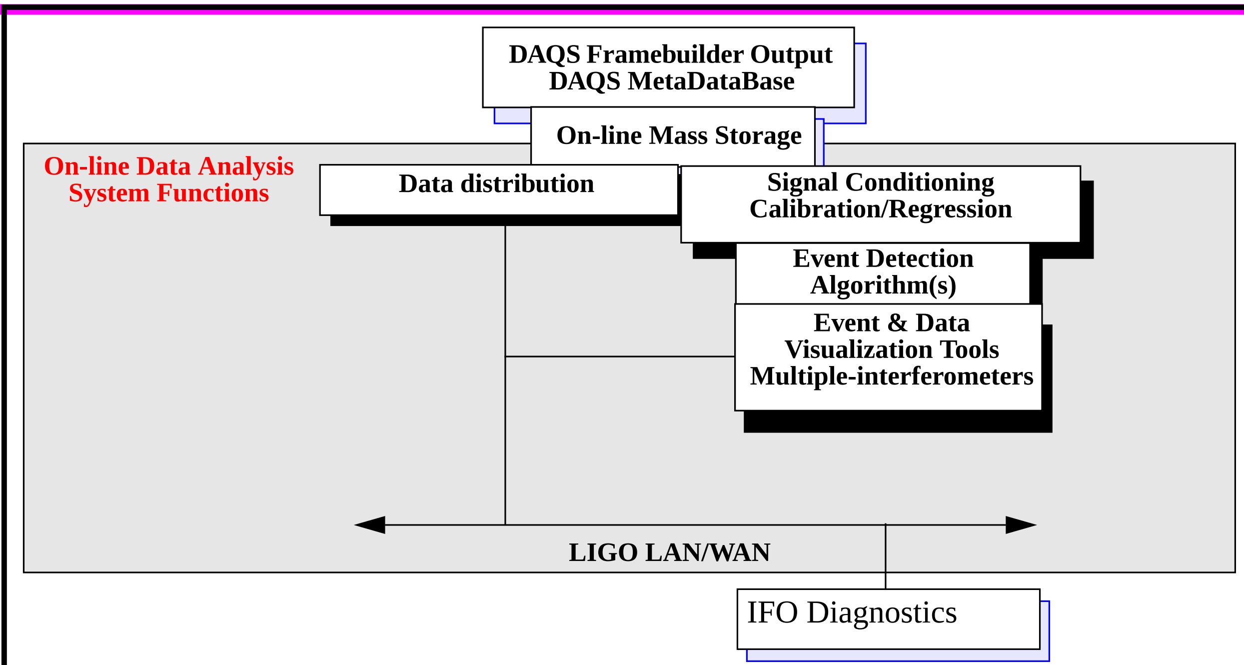
- signal processing for detection (continued)
  - shared candidate events
    - LIGO - LIGO
    - LIGO - VIRGO (?)
    - LIGO - GEO (?)
    - LIGO - EW detectors/observatories
- signal distribution
  - near real-time data available across LIGO Laboratory

- **Off-line component**
  - data ingestion/data reduction
    - process, reduce, and incorporate recent data into archive
  - data distribution
    - retrieve, filter, and deliver data to LIGO researchers
  - large scale searches
    - expanded searches for inspiraling binary events
      - lighter chirp mass ($m > 0.2\ M_{sun}$)
      - deeper (e.g., refined calibration, coherent processing of multiple interferometers, etc.)
    - extended periodic source searches
      - greater number of directions/solid angle coverage
      - larger parameter space

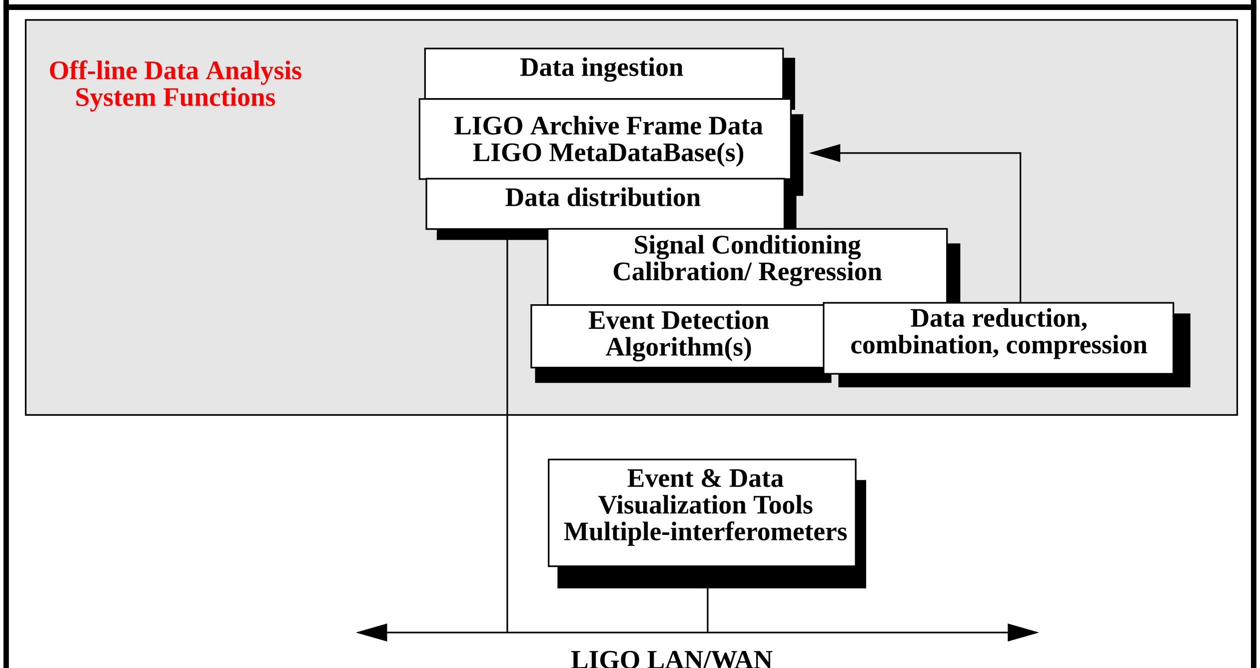
LIGO Data Analysis System (LDAS)
Functions

- Off-line component (continued)
  - non-time critical searches
    - stochastic background
    - data mining
    - refined analyses, etc.
  - algorithm prototyping and development
  - hardware upgrade/prototyping testbeds
  - repository of software developed across collaboration

LIGO Data Analysis System (LDAS)
Functional Interfaces



LDAS Block Diagram: On-line Analysis

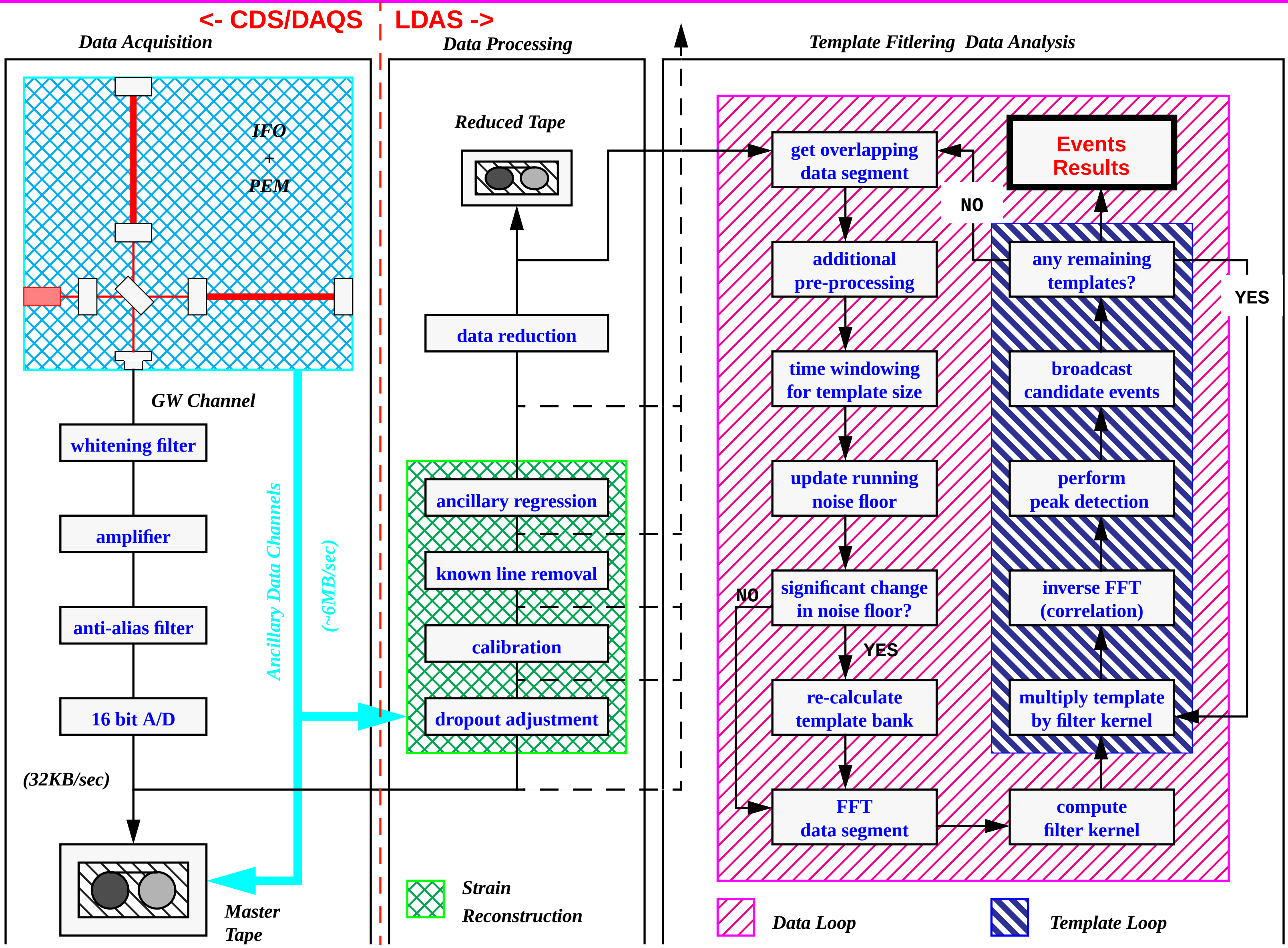


LDAS Block Diagram: Off-line Analysis

LIGO Data Analysis System (LDAS)

Data Analysis Flow Model -- On-line

Data analysis flow for optimal filtering with template detection



LIGO Data Analysis System (LDAS)
On-line system design

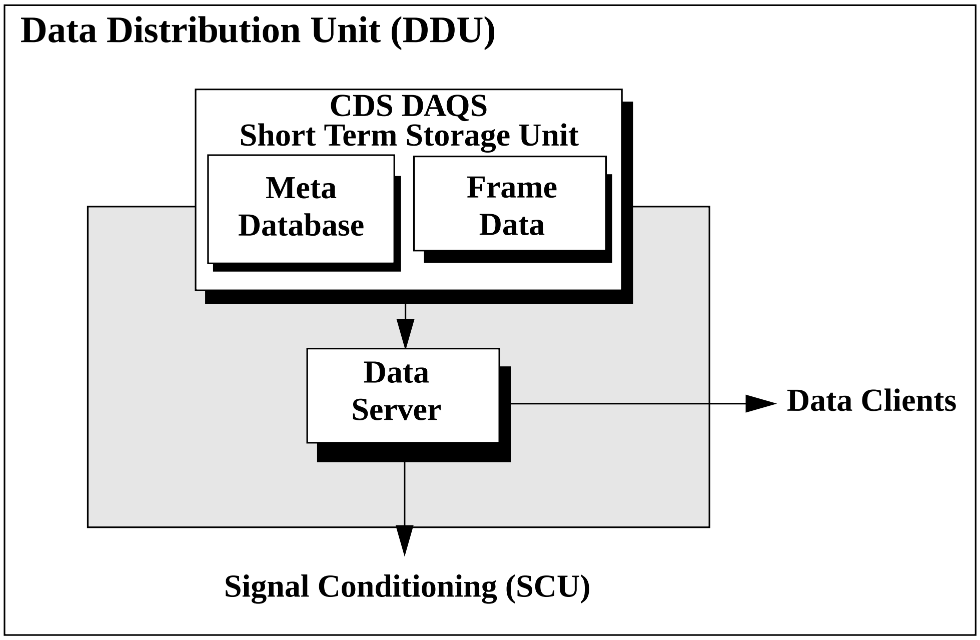
- On-line system implementation driven by the target analysis goals
- Most stressing: massively parallelized optimal matched filtering:
    - » Applicable to any source for which time-dependent signature can be derived
        - – Inspirals - NS/NS; BH/NS (?); BH/BH(??) -- breakdown of PN approximation may require other techniques (cf. below)
        - – BH ringdowns
        - – Transient: asymmetric SN (?) -- presently need other techniques until waveforms are available
    - » Have developed a detailed data analysis flow model to size the on-line systems
- System will also accommodate:
    - » diagnostics analysis;
    - » phenomenological signal processing;
    - » model-independent processing
        - – wavelets
        - – IFO $\otimes$ IFO
        - – ...

LIGO Data Analysis System (LDAS)
On-line system design

- Technology:
  - CPUs: Workstation/PC clusters => MPI/parallel computing
  - Data I/O: SCSI (ultra wide)
  - Communications: ATM/Fast Ethernet
  - Storage: Disk systems (RAID?) - fast cache for analysis
- Salient features (ref: Appendix A, T970160):

Process	*CPU*	*RAM*	*Disk*	*I/O*	*GFLOPS*
Inspiral Search	32 nodes X 300 MHz	128/Node	9 GB/node;	20+ MB/s disk cache; 12 MB/s MPI (100BT)	10
Data Conditioning [calibration + 64 line removal]	1 node X 300 MHz	11 MB	-	-	.010
Data Reduction [bandwidth reduction to 1 kS/S 32 channel regression w/ 1024 f0bins]	1 node X 300 MHz	.320 MB	-	-	.006
Totals	34 nodes	4 GB	270 GB	Ultra wide SCSI 100BT ATM(?)	10+

LIGO Data Analysis System (LDAS)
Data Distribution Unit - On-line

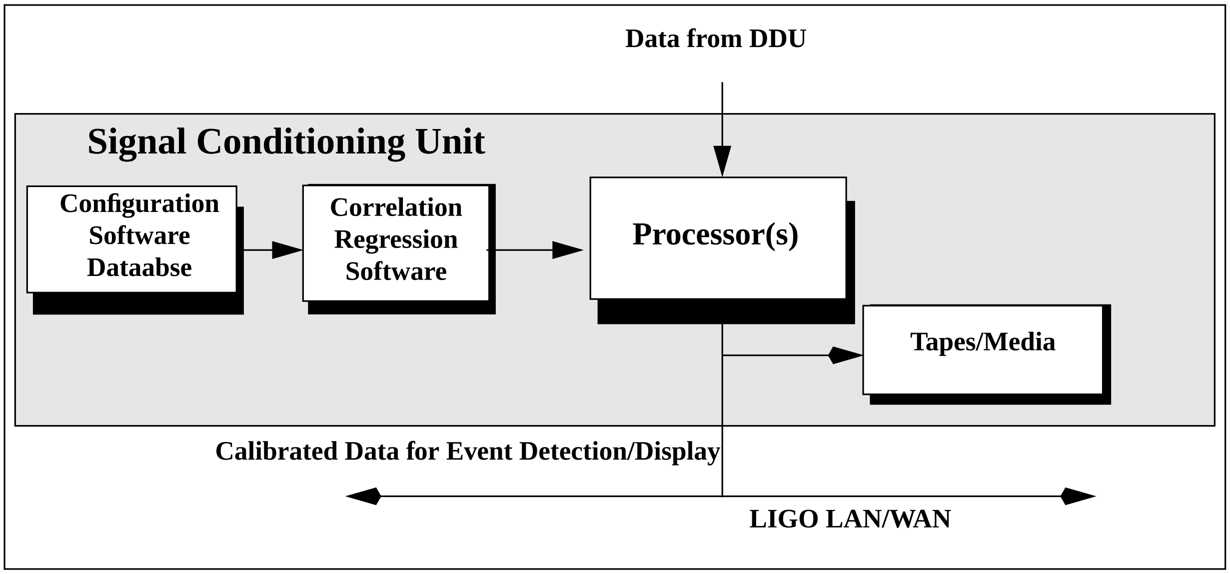


Data Distribution Block Diagram

Data Distribution Unit (DDU) -- On-line

Element	*CPU*		*I/O*	*STORAGE*
RAID System (shared with CDS/DAQS)	-	-	-	>500 GB striped(TBD)
Ultra Wide SCSI Port	-	-	> 40 MB/s	-
Data Server	>200 MFLOPS	>256 MB RAM	-	-
Network	-	-	- LIGO LDAS LAN (100BT/100Mb/s) - LIGO Site LAN (OC3/155Mb/s)	-

LIGO Data Analysis System (LDAS)
Signal Conditioning Unit - On-line



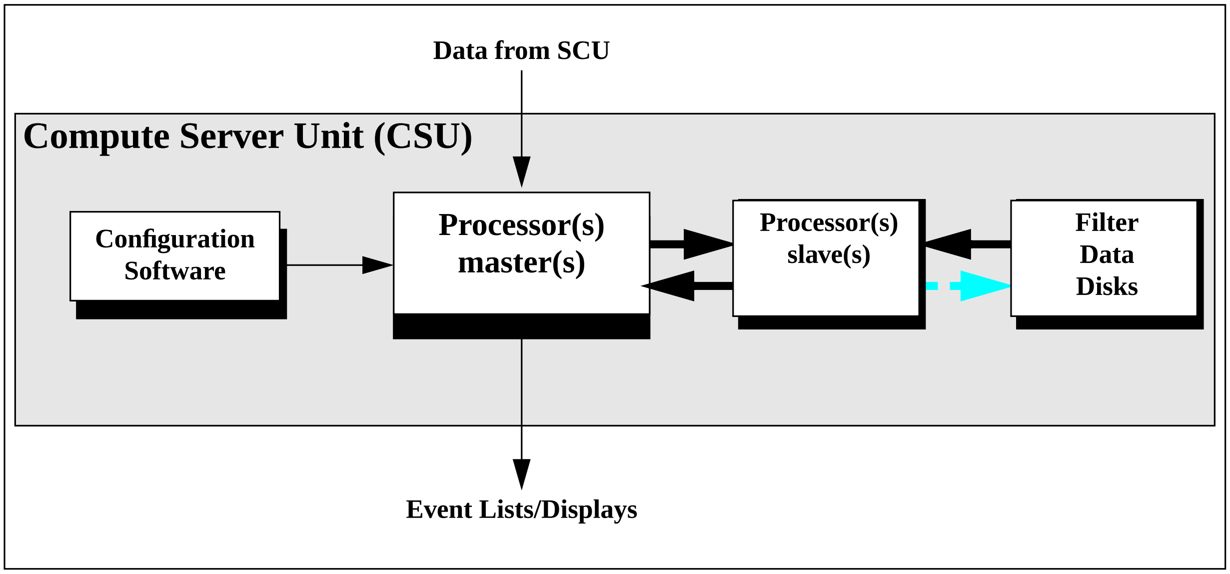
Signal Conditioning Block Diagram

.

Signal Conditioning Unit (SCU) -- On-line

Element	*CPU*		*I/O*	*STORAGE*
CPU	>200MFLOPS	>128 MB RAM	-	-
Ultra Wide SCSI Ports	-	-	> 40 MB/s 2 ports	-
Tape Drive/Robot	-	-	>2.5 MB/s >5 tape storage	>25 GB/tape ***CDs attractive @ .25 MB/s**
Network	-	-	- CSU direct (≥ 100BT) - LIGO LDAS LAN (OC3/155Mb/s)	-

LIGO Data Analysis System (LDAS)
Compute Server Unit - On-line



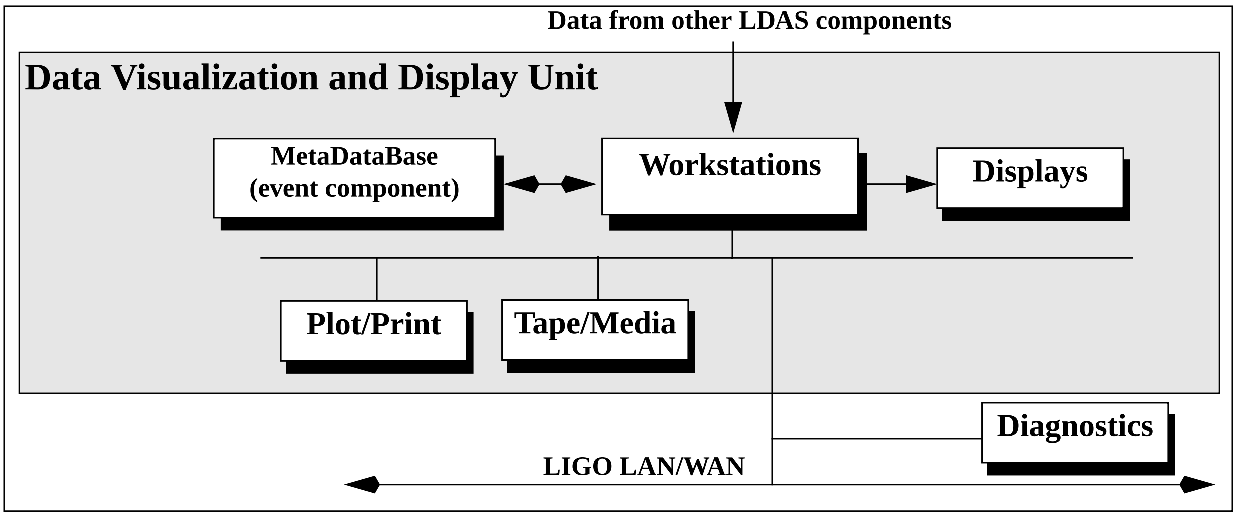
Compute Server Unit Block Diagram

.

Compute Server Unit (CSU) -- On-line

Element	*CPU*		*I/O*	*STORAGE*
CPU	32 nodes, $\geq$ 10 GFLOPS in aggregate	Per node, $\geq$ 128 MB RAM	100BT Ethernet	per node, 10GB disk
Fast Wide SCSI Ports	-	-	> 20 MB/s 1 port per node	-
Network	-	-	- LIGO LDAS LAN (OC3/155Mb/s) switched, point-to-point	-

LIGO Data Analysis System (LDAS)
Control/Monitoring Unit - On-line

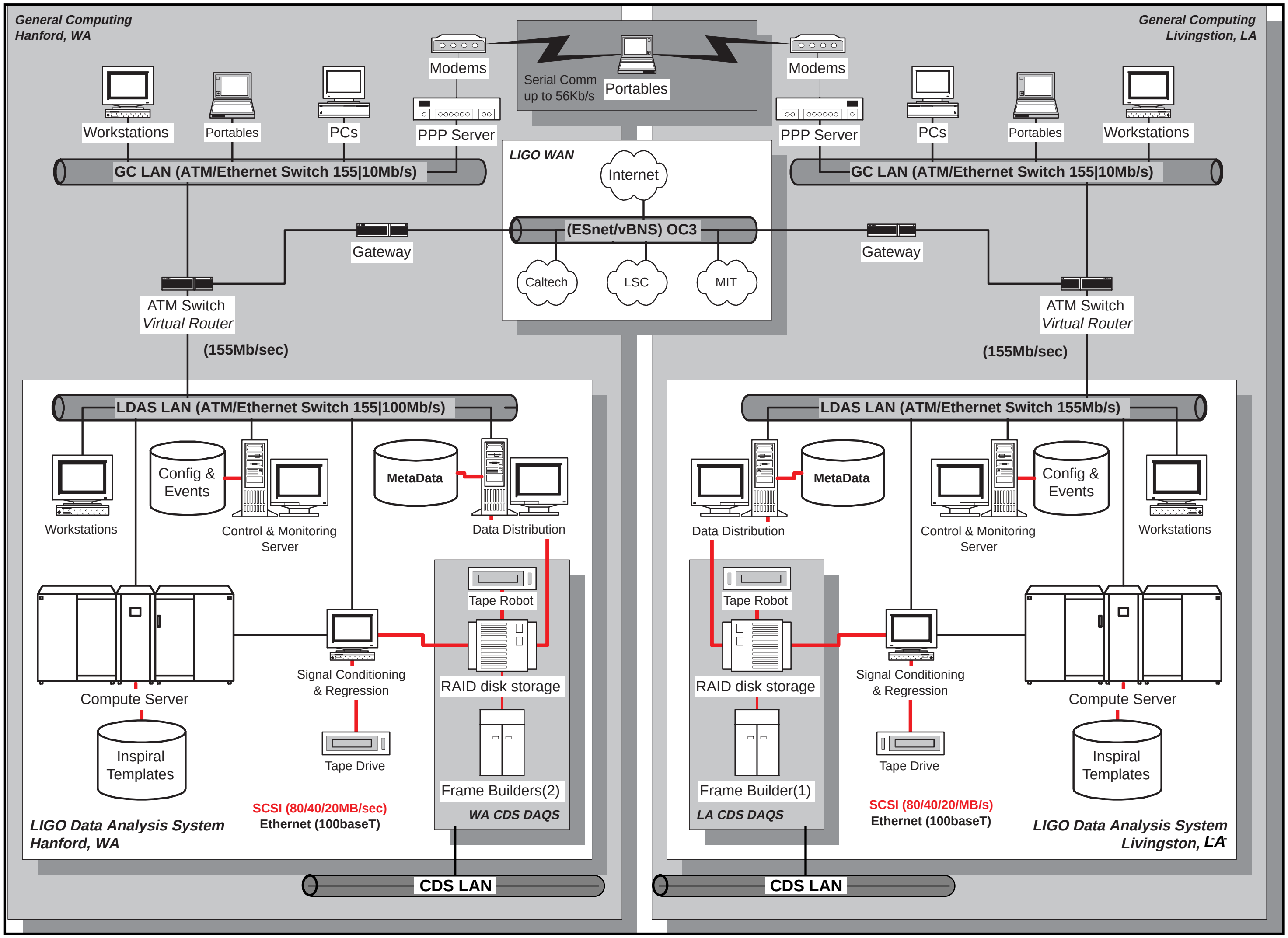


DVU Block Diagram

Control & Monitoring Unit (DVU) -- On-line

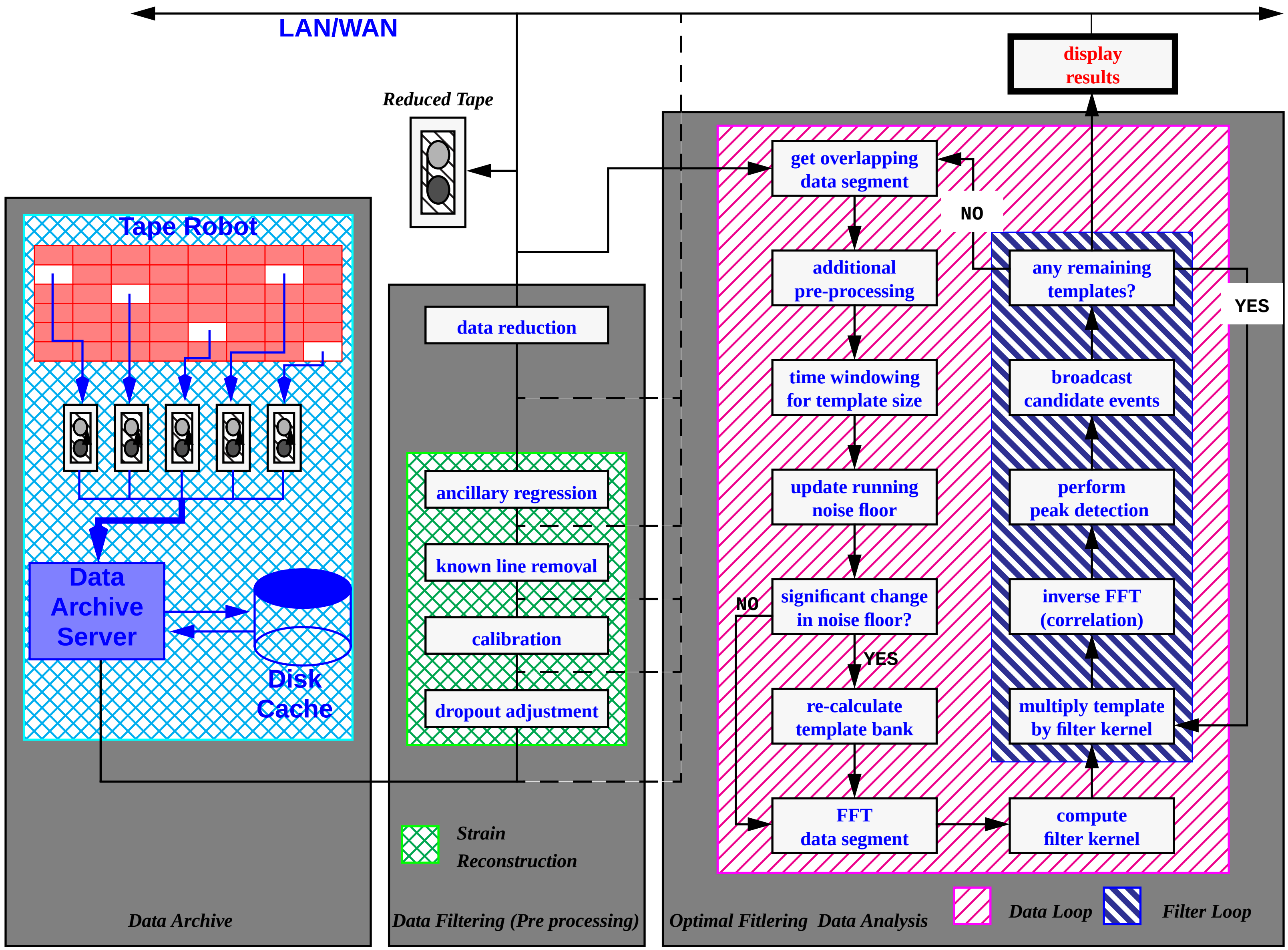
Element	*CPU*		*I/O*	*STORAGE*
CPU	>200MFLOPS /CPU $\geq$ 2 CPU	512 MB/ CPU	-	-
Ultra Wide SCSI Ports	-	-	> 40 MB/s 1 port	-
Disk	-	-	-	> 50 GB (TBD)
Network	-	-	- LIGO LDAS LAN (OC3/155Mb/s)	-

LIGO Data Analysis System
On-line component architecture



LIGO Data Analysis System (LDAS)
Data Flow Model -- Off-line

Data flow for off-line retrieval and analysis



LIGO Data Analysis System (LDAS)
Off-line system design

- Off-line system implementation driven by available (and rapidly improving) technology
  - Resource limited - available resources quickly accommodated...
- Two distinct off-line uses:
  - Data analysis/searches
    - Similar in complexity to on-line scope;
    - More computational demand (multiple users; deeper searches; etc.)
  - Data reduction/archival/distribution
    - Reduction not a computational issue (need algorithms/ discriminators)
    - Usage model not yet known
      - Fluctuating demand load
      - Span of epochs recalled
      - Length/volume of recalls
    - Provide a state of the art archival system (HPSS or similar)
      - Multi-head tape robot with tape cabinet
      - DVD jukebox may be an option for some data
      - Large volume disk cache
      - High bandwidth network for data transmission within CACR
      - Connection to vBNS for remote users

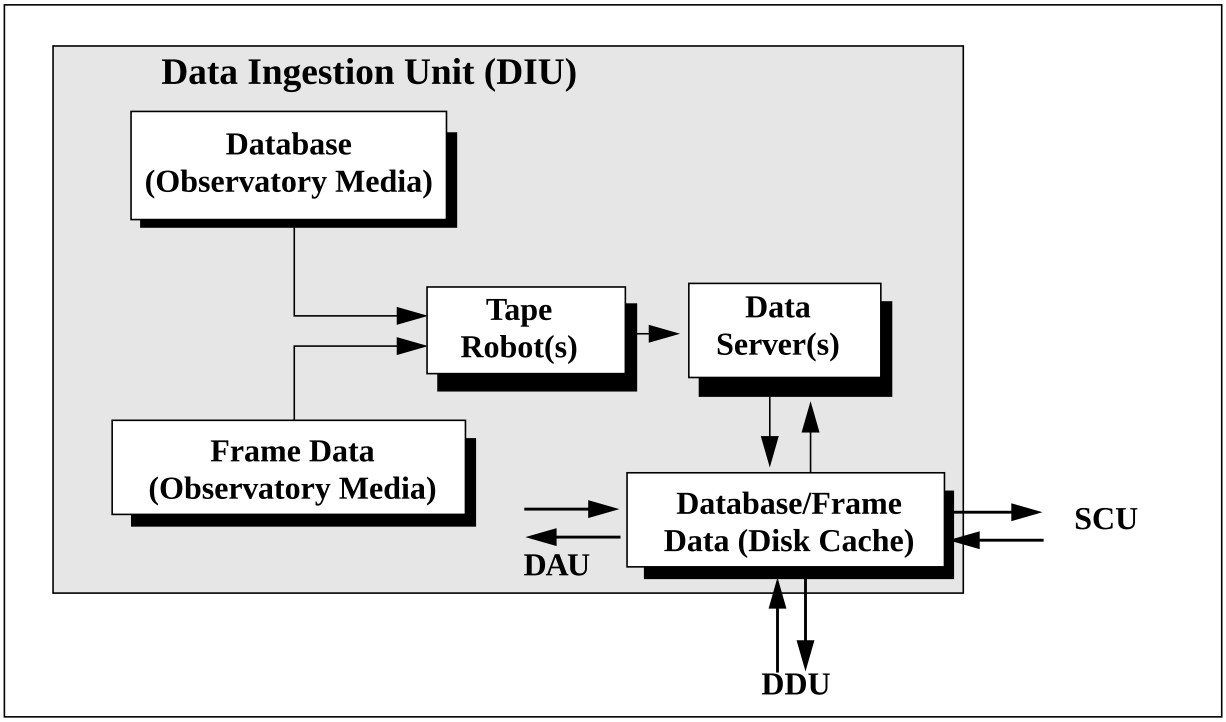
LIGO Data Analysis System (LDAS)
Off-line system design

- Data reduction/archival/distribution - continued
  - Data retrieved from tape will have a latency of minutes:
    - Locate and retrieve cassette
    - Mount cassette
    - Dump cassette onto disk cache (6 - 12 MB/s)
      - @ 40 GB/cassette => 1 hour read time
  - Data retrieved from disk will be limited by I/O bandwidth of technology:
    - 20/40/80 MB/s SCSI IO
    - 10 - 80 MB/s network speeds (between local computers for processing)
    - Long-distance transmission determined by vBNS access quality (up to ATM (peak) bandwidths)
    - => localize computation; distribute results

Data Ingestion (DIU) Block Diagram

- **Technology:**
  - CPUs: Workstation/PC clusters => MPI/parallel computing
  - Existing CACR SC resources
  - Data I/O: SCSI (ultra wide)
  - Communications: ATM/Fast Ethernet/vBNS (WAN)
  - Storage: Disk systems (RAID?) - fast cache for analysis

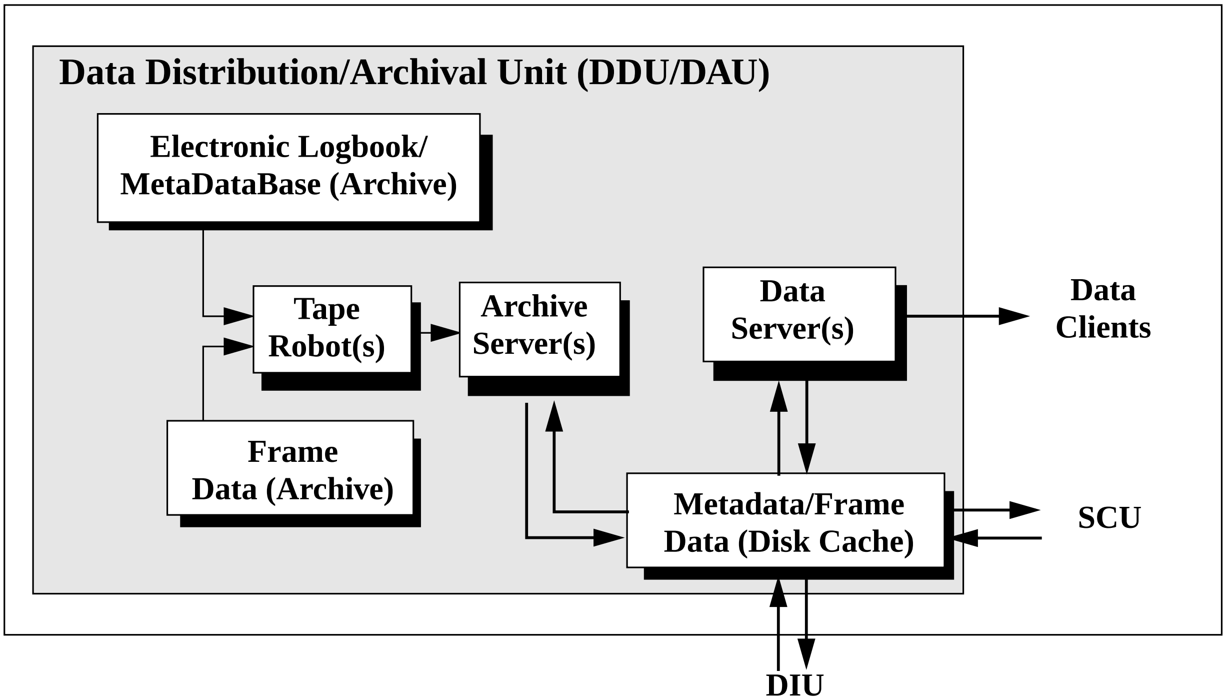
LIGO Data Analysis System (LDAS)
Data Ingestion Unit - Off-line



Data Ingestion Unit (DIU) -- Off-line

Element	*CPU*		*I/O*	*STORAGE*
Server/CPU	>200MFLOPS	512 MB		
Ultra Wide SCSI Ports	-	-	> 40 MB/s 1 port	-
Tape Drive/Robot	-	-	>6 MB/s, 3 head 10 tape storage	CDS compatible design
Disk Cache	-	-	40 MB/s, 4 ports	$\geq$ 250 GB (10 x 25GB/tape) Striped for 3 users (3 IFOs) Partition on RAID shared with DAU/DDU

LIGO Data Analysis System (LDAS)
Data Distribution/Archival Unit - Off-line



Data Distribution/Archival (DDU/DAU) Block Diagram

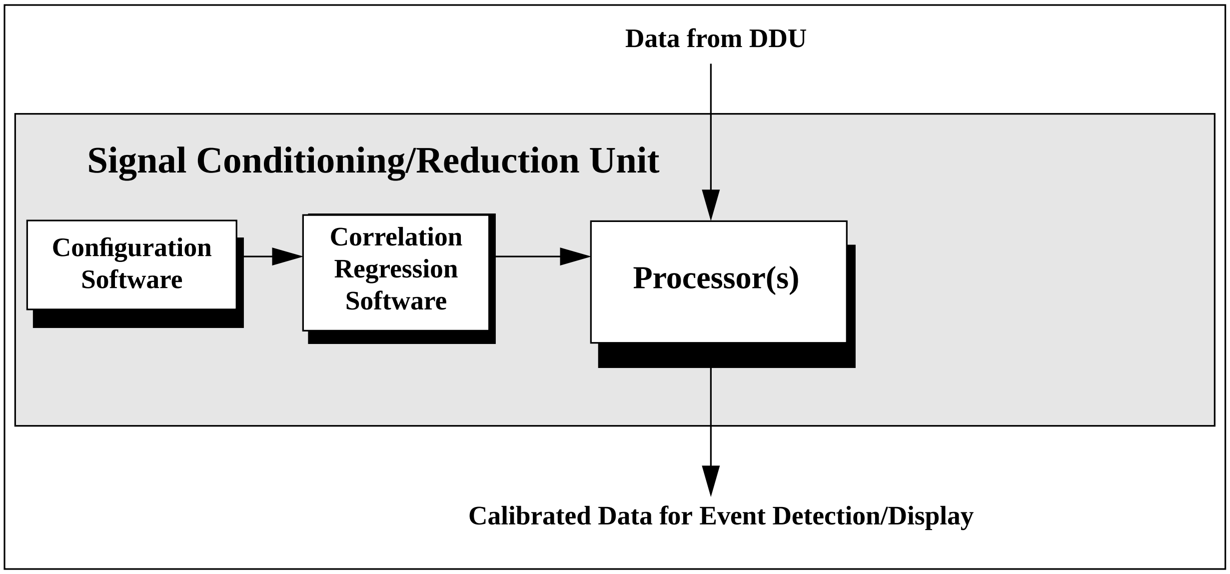
Data Distribution Unit (DDU) -- Off-line

Element	*CPU*		*I/O*	*STORAGE*
Data Server	2 node, >200 MFLOPS per node	≥ 1 GB RAM		-
Disk System	-	-		Common with DAU
Fast Wide SCSI Port			>40 MB/s 1 port	
Network access	-	-	- CACR LDAS LAN (OC3/155 mb/s) - HiPPI	-

Data Archival Unit (DAU) -- Off-line

Element	***CPU***		***I/O***	***STORAGE***
Tape Robot & Cabinets			≥ 5 Tape heads 6 MB/s	≥ 100 TB 3 cabinet (2 years @ 10% “online”)
Disk System (Shared with DDU)	-	-	HiPPI	>500 GB striped for 5 users =4 * #users * vol/tape (25 GB) Partition on shared resource with DDU/DIU
Archive Server	2 node, >200 MFLOPS per node	≥ 512 MB RAM	HiPPI	-
Network	-	-	- CACR LDAS LAN - CACR HiPPI	-

LIGO Data Analysis System (LDAS)
Signal Conditioning Unit - Off-line



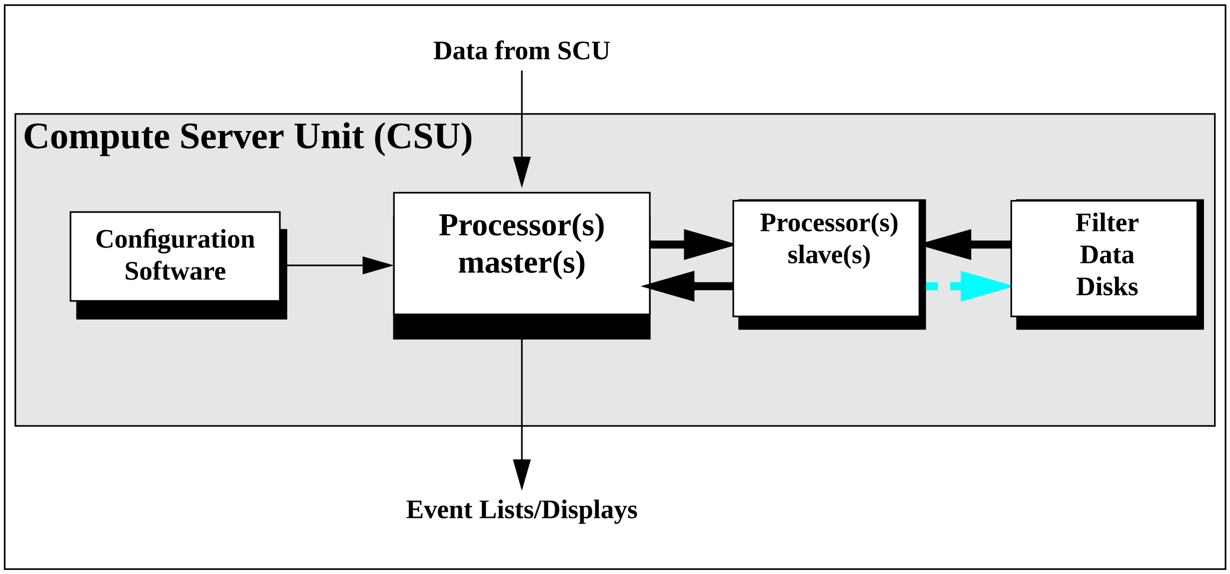
Signal Conditioning Block Diagram

.

Signal Conditioning Unit (SCU)-- Off-line

Element	*CPU*		*I/O*	*STORAGE*
CPU	>200MFLOPS/CPU ≥ 6 CPUs	1 GB RAM	-	-
Ultra Wide SCSI Ports	-	-	> 40 MB/s 2 ports	-
Tape Drive/Robot	-	-	>5 MB/s 5 tape storage	>25 GB/tape
Network access	-	-	- LIGO LDAS LAN (OC3/155Mb/s) - CSU direct (100BT, 100 Mb/s)	-

LIGO Data Analysis System (LDAS)
Data Reduction/Compute Server Unit - Off-line

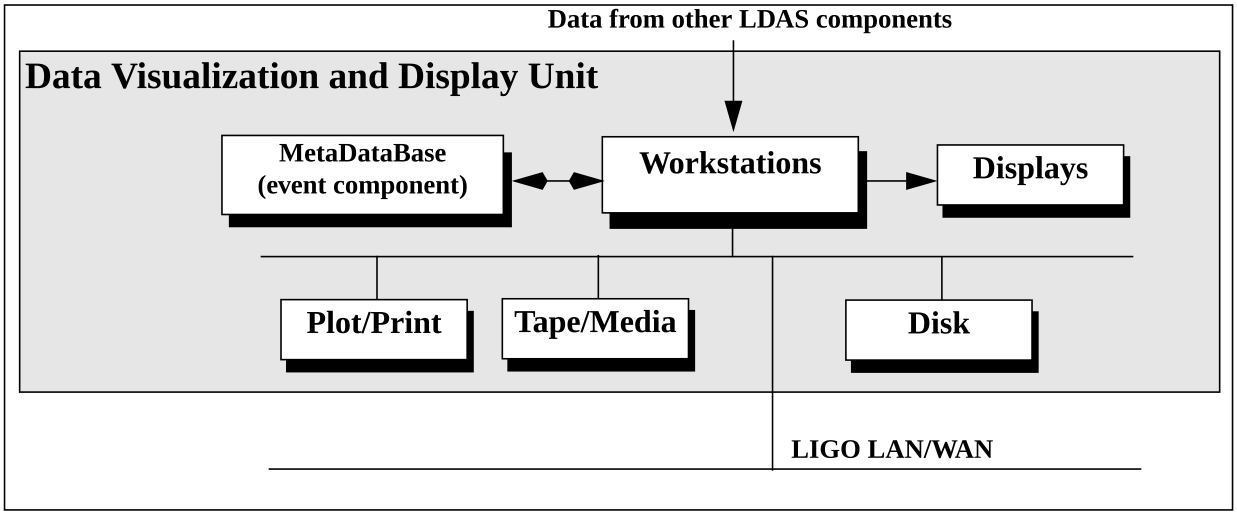


Compute Server Unit Block Diagram

Compute Server Unit (CSU) -- Off-line

Element	*CPU*		*I/O*	*STORAGE*
CPU	96(TBD) nodes, aggregate 30 |GFLOPS	Per node, 256 MB RAM	100 BT (100 Mb/s)	per node, 10 GB disk
Fast Wide SCSI Ports	-	-	> 20 MB/s 1 port per node	-
Network access	-	-	- LIGO LDAS LAN (OC3/155Mb/s)	-

LIGO Data Analysis System (LDAS)
Control & Monitoring Unit - Off-line

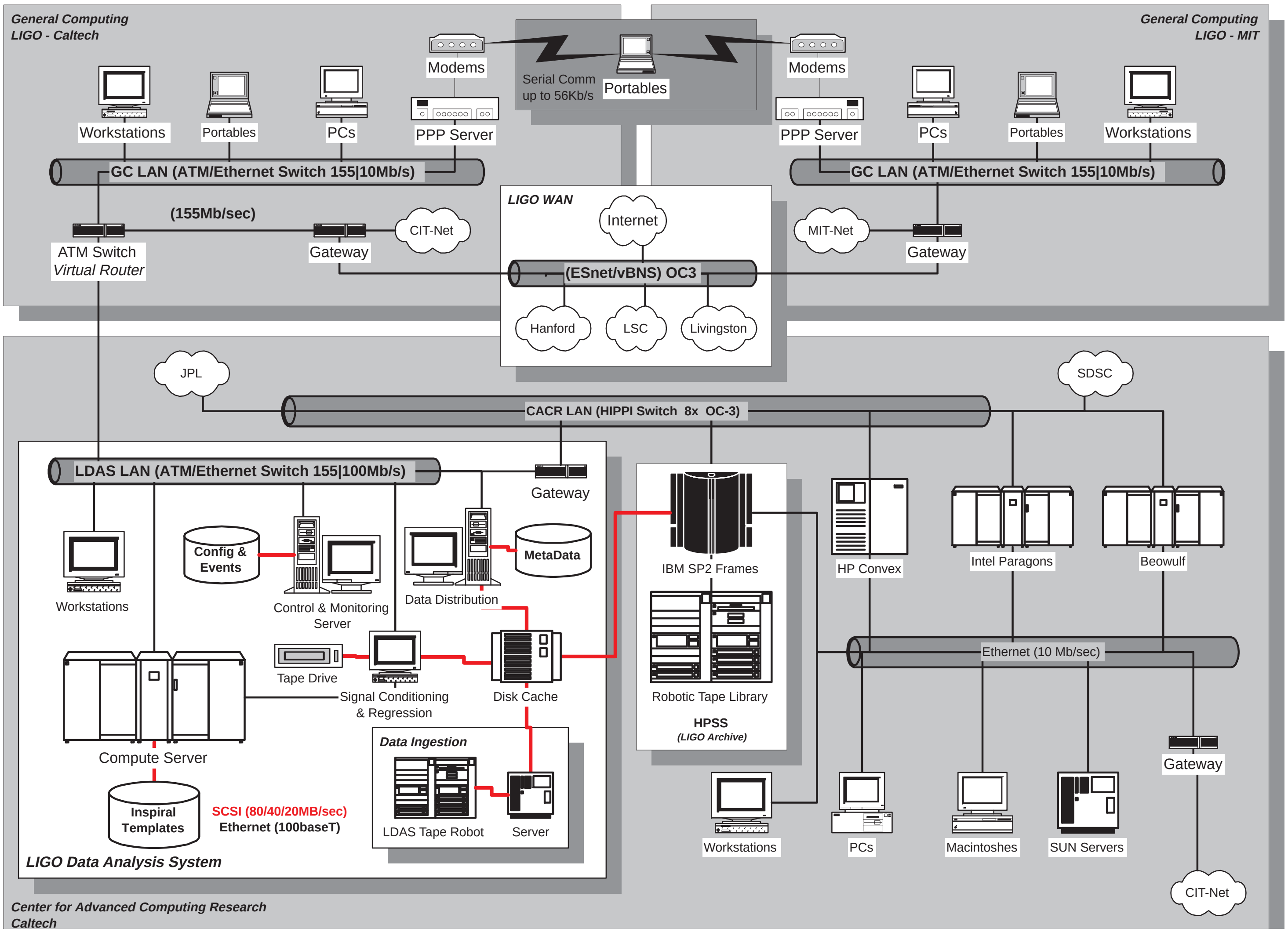


DVU Block Diagram

Control & Monitoring Unit (DVU)-- Off-line

Element	*CPU*		*I/O*	*STORAGE*
CPU	>200MFLOPS/CPU ≥ 4 CPUs	2 GB RAM	-	-
Ultra Wide SCSI Ports	-	-	> 40 MB/s 2 ports	-
Disk	-	-	-	>50 GB
Network access	-	-	- LIGO LDAS LAN (OC3/155Mb/s)	-

LIGO Data Analysis System
Off-line Component Architecture



PC SUPER CLUSTERS

BEOWULF PARADIGM

LIGO proposes a Beowulf

Roy Williams

Center for Advanced Computing Research
California Institute of Technology

A Beowulf machine

is a collection of ordinary PC's with Pentium Pro, Pentium II, etc
with off-the-shelf networking 100 Mbit/sec
running free software such as Linux, GNU, X

- It is really cheap
- very robust
- high megaflops
- high latency and medium-speed communication
- not plug-and-play
  - --but neither is any Unix system
- reasonably scalable

Projects include

- CDF experiment at Fermilab
  - Connecting a "pcfarm" to VxWorks via ATM connection
- European Southern Observatory
  - online processing for Chile telescope
- Many at Caltech
  - Environmental engineering, Chemistry, JPL flight emulator
- Italian Space Agency
  - on-demand SAR processing
- 140-processor system Caltech/CACR
- 500-processor system at NASA Ames (soon)
- Sandia (248), Goddard (190),
- many other systems
  - Drexel, George Mason U, UC Berkeley, MIT, U Washington, ...

Beowulf is

- Cluster of PCs / Pile of PCs / PCFarm
- Pure mass-market COTS
- All free software
- Unix OS with source code availability (Linux / BSD Unix)
- A message-passing parallel computer with MPI and PVM
- Best price-performance
- Rapid response to technology trends
- No single point vendor
- Leverages software development from CS community
- Mature, robust, accessible

Beowulf for LIGO?

- How?
  - Online system at Hanford/Livingston
  - Intelligent spinning archive (“Datawolf”)
- Why?
  - Few, controlled applications
    - (online processing, data serving)
  - Low communication
  - CACR connection
- Portability
  - If code runs on Sun and Pentium, it must be portable

Beowulf Cost

Table 1: Gigaflop Beowulf Supercomputer, October 1997

Venus motherboard 200 MHz Pentium Pro	$750	16	$12K
32 Mbyte SIMM EDO 60 nsec memory (=2 Gbyte)	$115	64	$7K
3.2 Gbyte disk (=100 GByte)	$225	32	$7K
Bay networks 350T fast ethernet	$2600	1	$3K
Video monitor 21"	$2000	1	$2K
Packaging	$86	1	
Misc			$4K
Total			**$36K**

LIGO Benchmark

B. Allen, U of Wisconsin
ballen@dirac.phys.uwm.edu

Searching for binary inspiral events,
Wiener (optimal) filtering to detect short templates in data
Very parallel: we look for each template in each data segment.

Table 1: Number of CPU's to keep up with data stream

	small	medium	full Ligo (extrapolated)
Intel Paragon	4	33	205
SGI Origin	1	3	90
Sun Ultra2		3	116
Beowulf Pentium Pro	2		96

Optimized FFT: Kuck for Paragon, SGI/Cray, Sun Performance, FFTW for Beowulf

All of these systems can provide
optimizing compilers and high quality FFT libraries

We believe the code can be improved significantly,
but it should be an overall scale on these results, with little relative change.

We believe these results to be representative for
generic problem of searching for short templates in signals.

Note: We expect **Pulsar Search** to present very different results:
the communication fabric is heavily stressed by gigapoint FFT's.

"Beowulf is the best machine around for LIGO
... and you can quote me on that"

Tom Prince
LIGO scientist

Note -- PC Clusters aggravate some problems

Significant node-to-node latency (~100 μsec)

Limited by cost, software

Will get better with new OS put-aside communications

Bandwidth limitations

(10 Mbit/sec, 100 Mbit/sec, 1000 Mbit/sec....)

Limited by cost

Will get better faster

LDAS DRR

December 12, 1997

LIGO DATA ANALYSIS SOFTWARE

LDAS Software Requirements

- LDAS Requirements are applicable to both hardware and software
- Software Specific Requirements:
  - ›› Portability:
    - Portable Operating System Interface compliant (POSIX) on Unix Platforms
    - ANSI Languages Compliant Code (C++ Standard, 11/14/97! - http://www.research.att.com/~bs/iso_release.html)
  - ›› Extensible:
    - Object Oriented Programming Techniques in C++
    - Modular, Reusable Code Units elsewhere
    - Distributed Computing based on MPI
  - ›› Maintainability:
    - Source Code Management using Concurrent Version System (CVS configured in client-server mode using CVSH)
    - Expressly Coded in Object Oriented C++ Language whenever possible
    - Keep It Simple Style (KISS) Guidelines for Coding Constructs
  - ›› Flexibility:
    - Object Oriented Design (C++)
    - Modular Libraries (C, C++, others: e.g Fortran...)
    - Centralized Server-Client(s) paradigm for program control
    - Remaining infrastructure based on Standard Libraries (STL)

LDAS Software Components

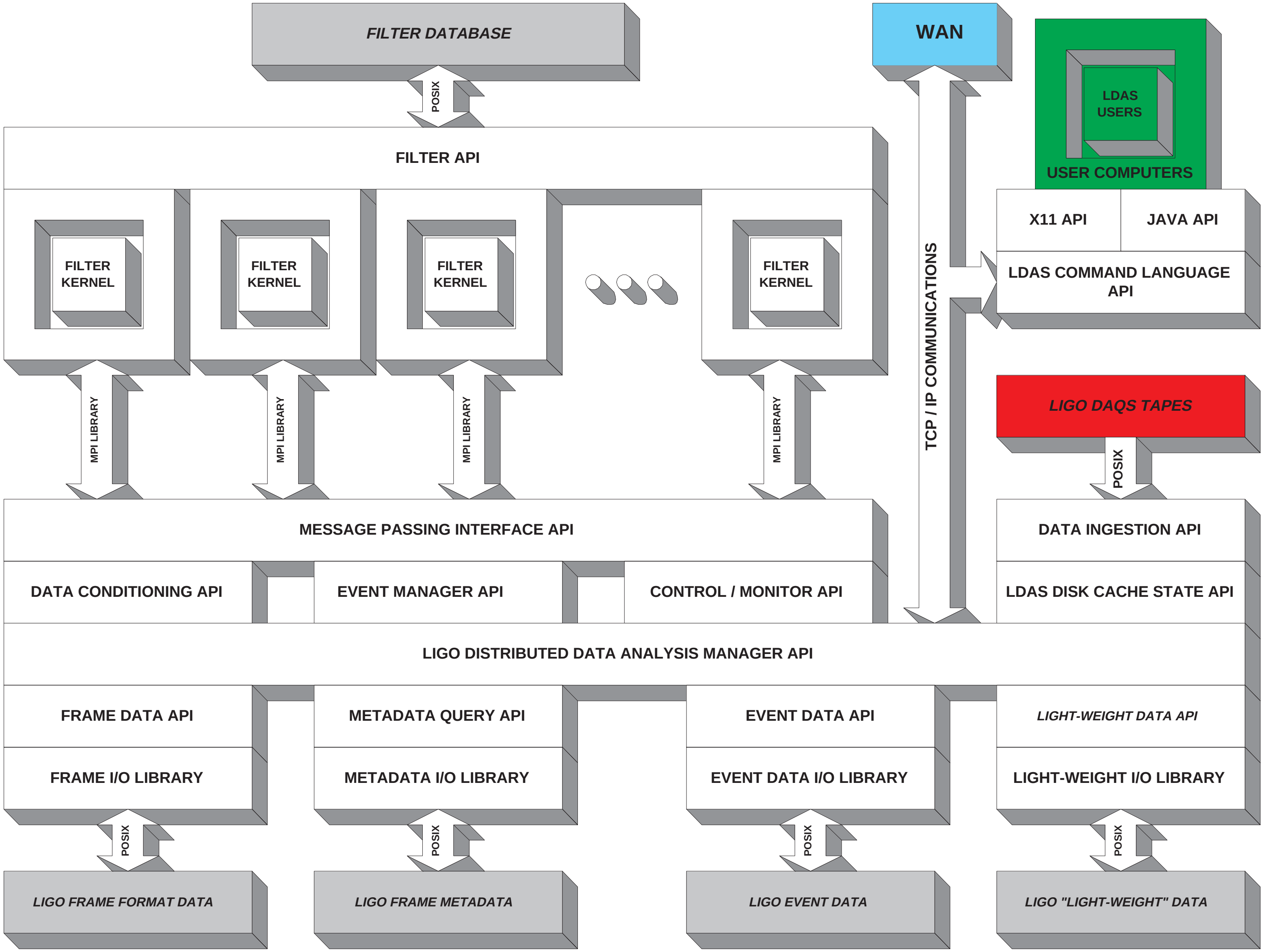
- Data Format Types:
  - » DAQS Tapes (Ingestion Process)
  - » Frames, Metadata(?), Lightweight(?), Event(?), Templates(?)
  - » LDAS Interprocess Data Format (Lightweight Format?)
- Low Level I/O Libraries using POSIX
  - » FrameLib, MetaLib, EventLib, LightWeightLib
- Application Programmer Interfaces (data flow management)
  - » High Level interfaces to all I/O Libraries: FrameAPI, MetaAPI, EventAPI, LightWeightAPI
  - » Control/Manage Layers: Command(Query) Language, Data Conditioning, Data Caching, Candidate Event Management, System Control & Monitoring
  - » Filtering Layers: Filter Control, Filter Database, Filter Generation, Message Passing, Data Distribution and Analysis Management

LDAS Software Components

- User Interfaces
  - X11 based windows (providing LAN connectivity)
    - Unix wide standard (Used everyday in LIGO!)
      - graphics computation left on host
      - network bandwidth consumed to draw windows
  - JAVA applets (providing WAN connectivity)
    - highly popular - emerging web standard
    - preferred solution due to optimal efficiency in use of CPUs
    - not ANSI standard - long term compatibility concerns
  - Command Language Parsing (Shell/Script Environment)
    - Sits under both the X11 and JAVA interfaces
    - efficient text-based communication
    - Integrates with telnet and code development tools
- Software Components layered over Unix Operating System, POSIX APIs and Unix filesystems!

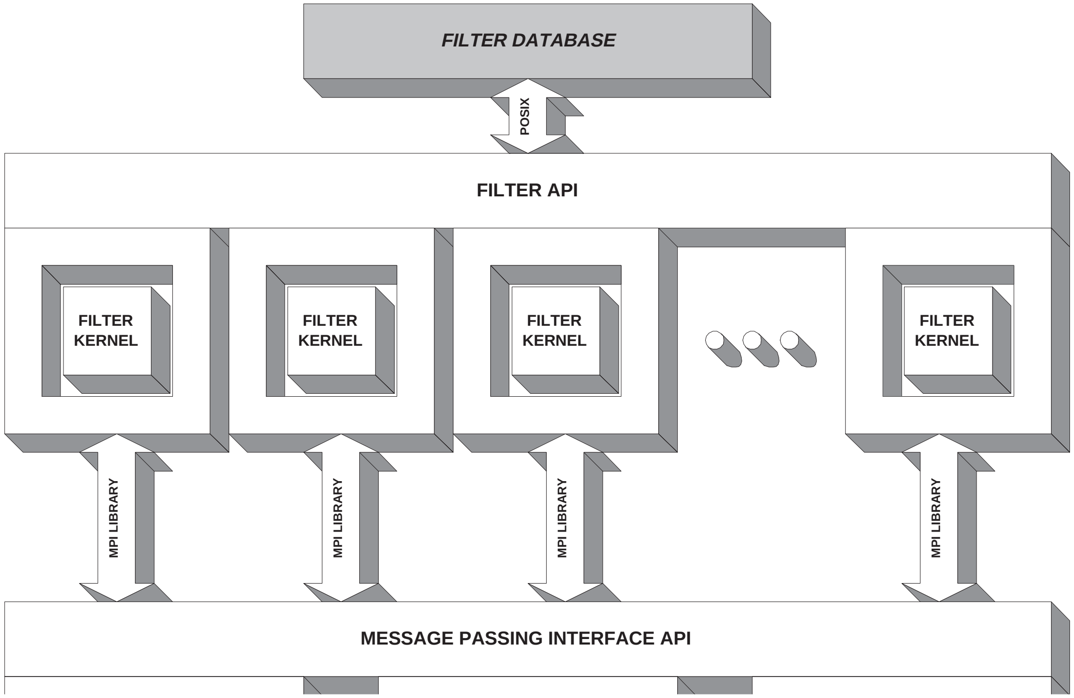
LDAS Software

Block Diagram of the LDAS Software Components



LDAS Number Crunching Software

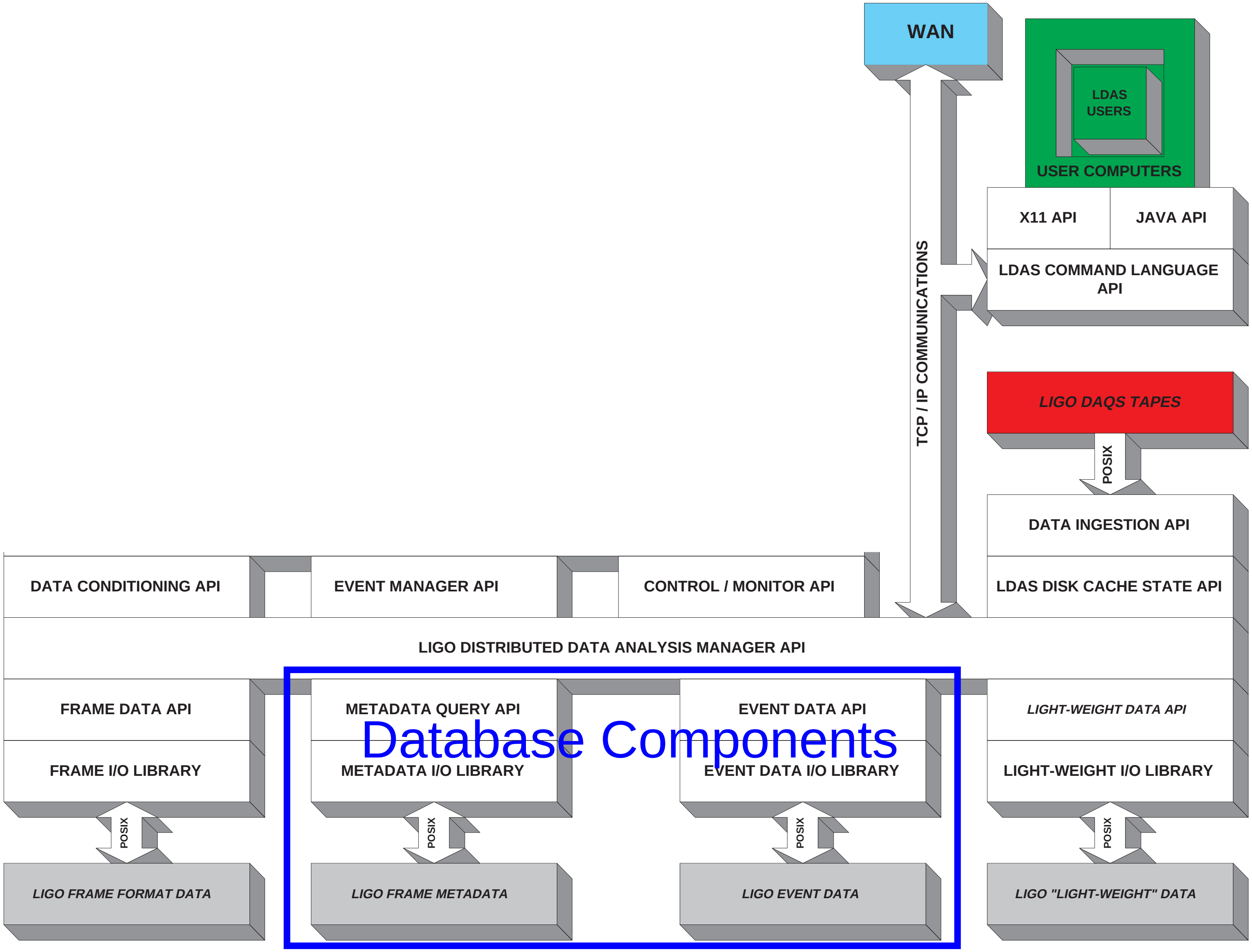
Block Diagram of the LDAS Software Components



Diagnostics
Binary Inspiral Searches
Instrumental Signitures
Time-Frequency Filters
...

LDAS Data Shuffling Software

Block Diagram of the LDAS Software Components



Data Formats

- FRAMES - LIGO/VIRGO Common Data Format
  - Primary Purpose: Data Acquisition / Data Archival
  - FRAME I/O Library Version 3.42 Specified (T970130-B)
  - Migration to separate I/O & API layers under C++
- Meta-Data - Data about Data (databases)
  - Specific Queries TBD -> leading to specific approach (commercial DB or OO?)
- Event Data - Candidate GW Detections
  - Also Data about Data
  - Narrower Scope of Queries (commercial DB or OO?)
- Light-Weight Data
  - Primary Purpose: Easy of Use
  - Candidates include Net-CDF, SDF, ASCII text
  - Other possible uses include LDAS interprocess data
  - Translators into popular Analysis Packages like Matlab, IDL, Mathematica needed
- Template Data
  - Waveforms for optimal filtering “pre-stored” on disk
  - Instrumental transient catalogs
  - Candidates: Light-Weight Format or MPI data-types

FRAME DATA FORMAT

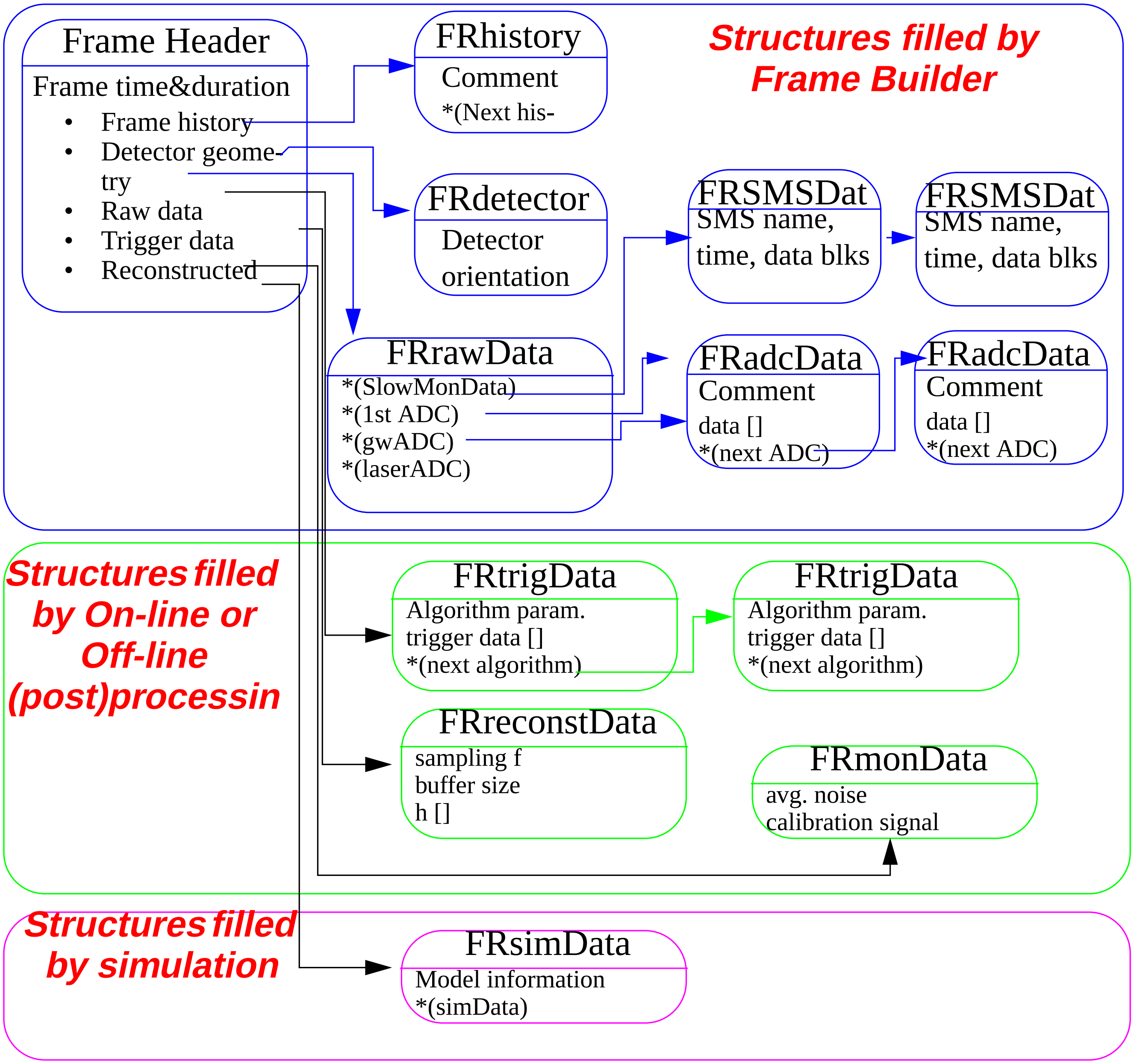
Frame Architecture

- Specification (T970130-B) jointly approved by LIGO and VIRGO in September
  - ›› FILES, with a file header, contain:
  - ›› FRAMES (unit of information containing all information needed to understand the interferometer behavior over a finite time interval) with a frame header, which contain:
  - ›› STRUCTURES (frames are organized as a set of C-like structures)
- Data classes defined in a platform-independent implementation:
  - ›› INT_2U => unsigned 2 byte integer (int, short, etc. generally mean different things on different machines).
  - ›› insensitive to byte-ordering on hardware
- Data objects (structures) presently implemented:

•Frame Header Structure
•Detector Data Structure
•Message Log Data Structure
•Frame Raw Data Structure
•Frame Simulated Data Structure
•Static Data Structure
•Frame Trigger Data Structure
•End of File Data Structure
•Frame ADC Data Structure
•End of Frame Data Structure
•Frame History Structure
•Post-Processed Data Structure
•Frame Serial Data Structure
•Frame Vector Data Structure
•Frame Summary Data Structure

FRAME DATA FORMAT

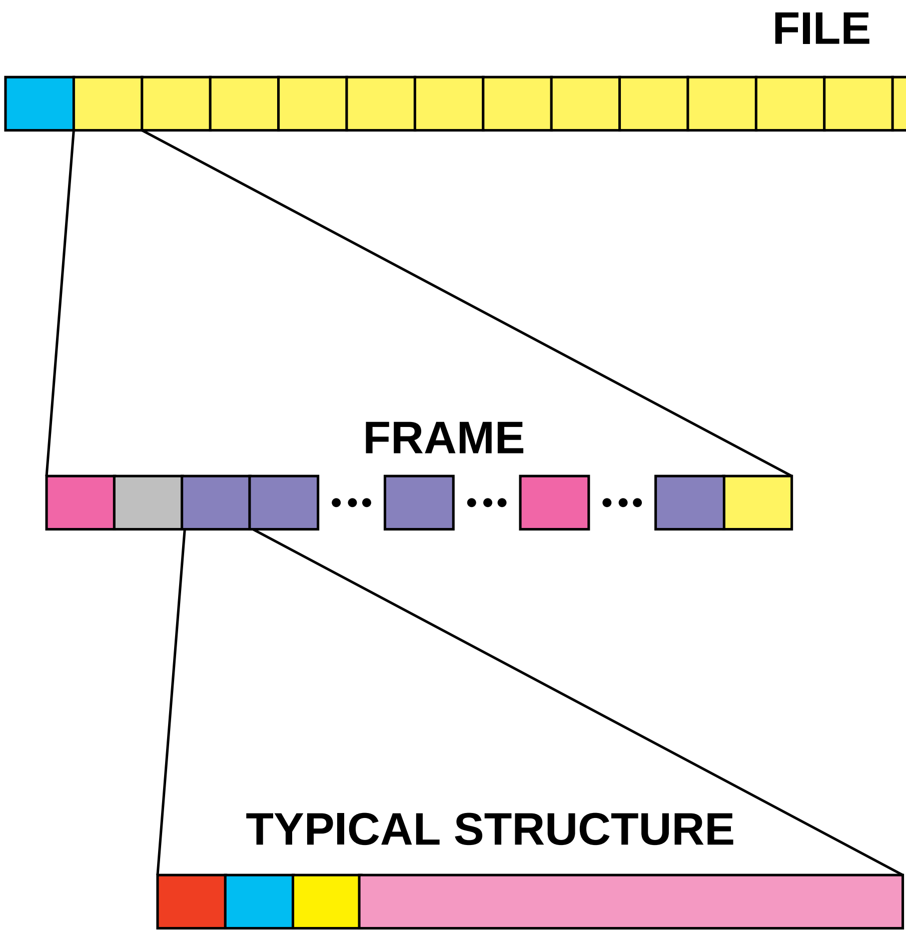
Structural inter-relationships



- **Frame has tree-like structure:**
  - **Individual blocks are C-like structures**
  - **Extensible to arbitrary length with design evolution**
  - **Utilized for both on-line & off-line analyses**

FRAME DATA FORMAT

Compositional relationships



FILE MAKE-UP

- FILE HEADER (1 per file)
- FRAME
- END OF FILE (1 per file)

FRAME MAKE-UP

- FRAME HEADER (1 per frame)
- DICTIONARY*
- DATA CLASSES
- END OF FRAME (1 per frame)

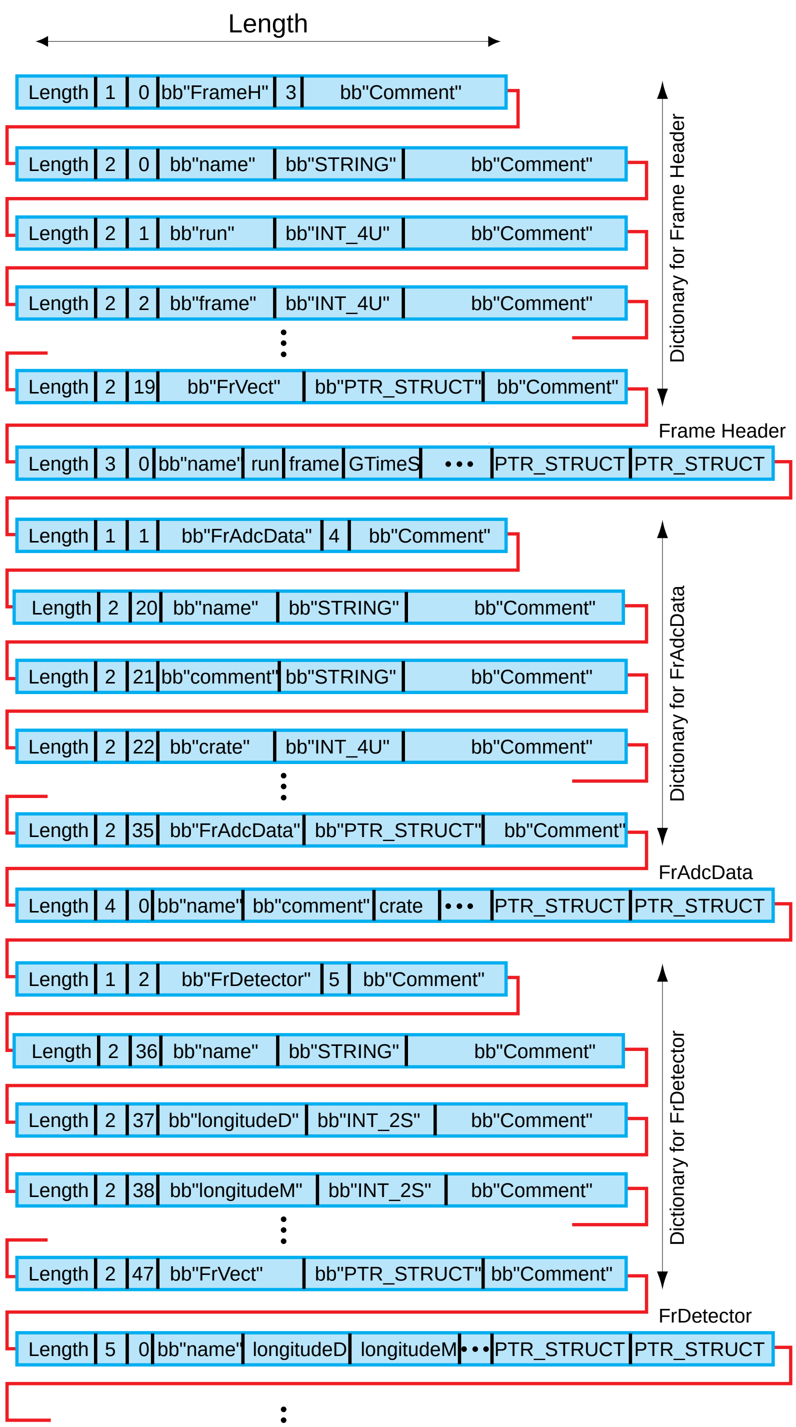
STRUCTURE MAKE-UP

- LENGTH
- DATA CLASS
- COUNTER FOR INSTANCE OF CLASS IN FRAME
- AGGREGATE DATA (VECTOR W/ VARIABLE TYPES)

*** Dictionary structure behavior is unique in that:**
1. It preceeds header for first frame of file;
2. Dictionary is built up incrementally as addititional structures are incorporated into frame
3. It is valid for entire file (persistent)

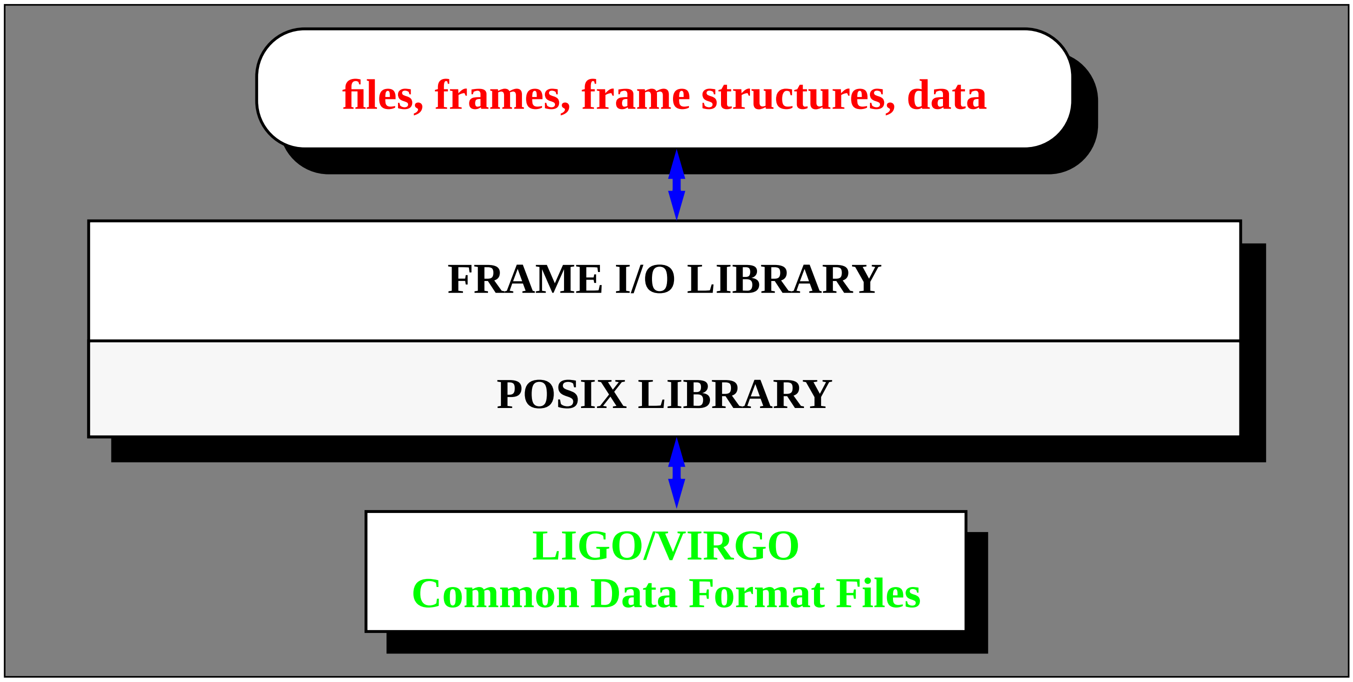
FRAME DATA FORMAT

Byte-level description



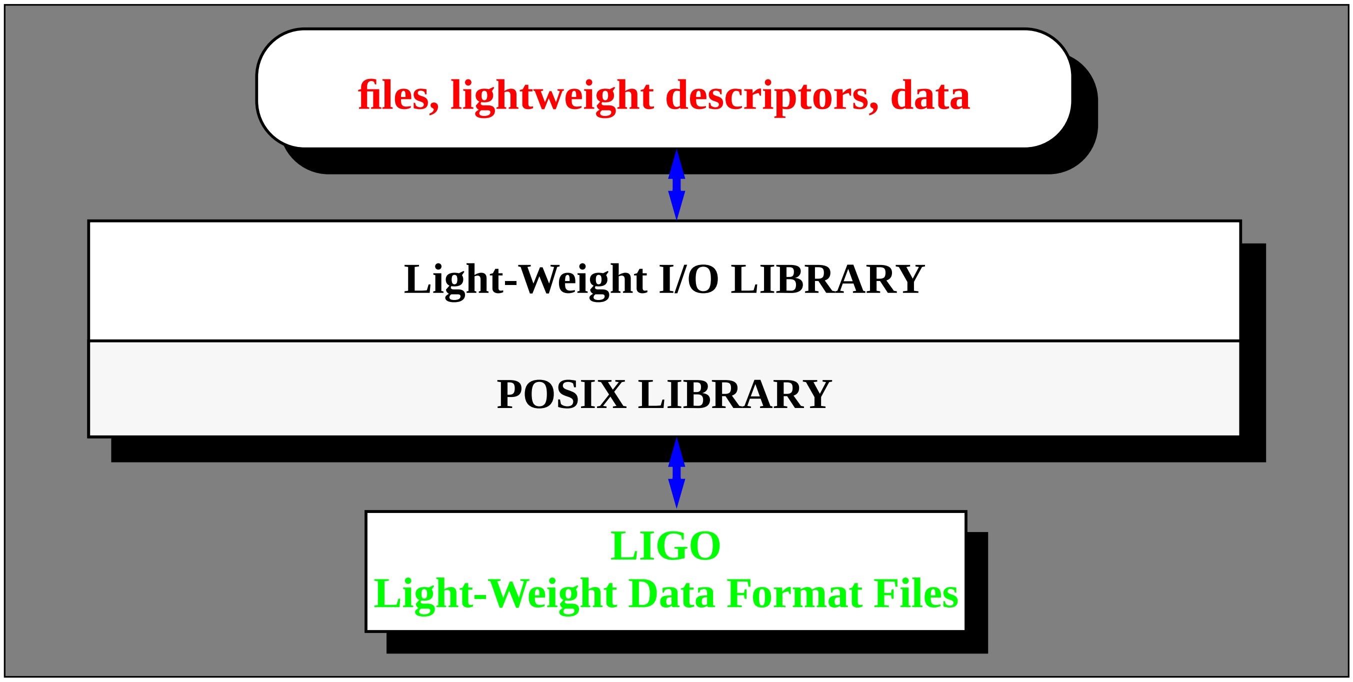
* bb"..." denotes a composite STRING type object defined in data type table
* refer to structure definition tables in text for byte length of various objects above.

Frame I/O Library



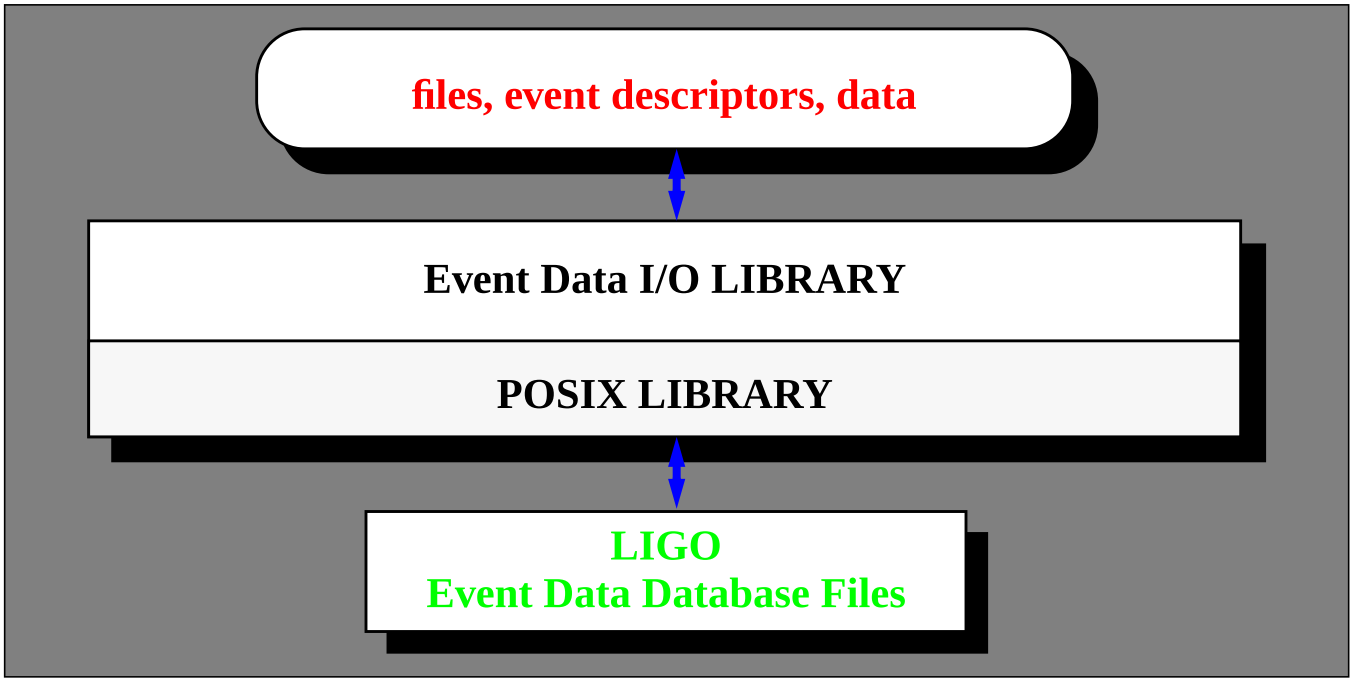
- Data:
  - Files, Frames, Frame Structures, Frame Structure Descriptors, Data
- Methods:
  - Read / Write Frames
  - Add / Extract Frame Structures into/from Frames
  - Add / Extract Frame Structure Descriptors “
  - Add / Extract Data “
  - Modify Frame Components
  - Error Handling

Light-Weight I/O Library



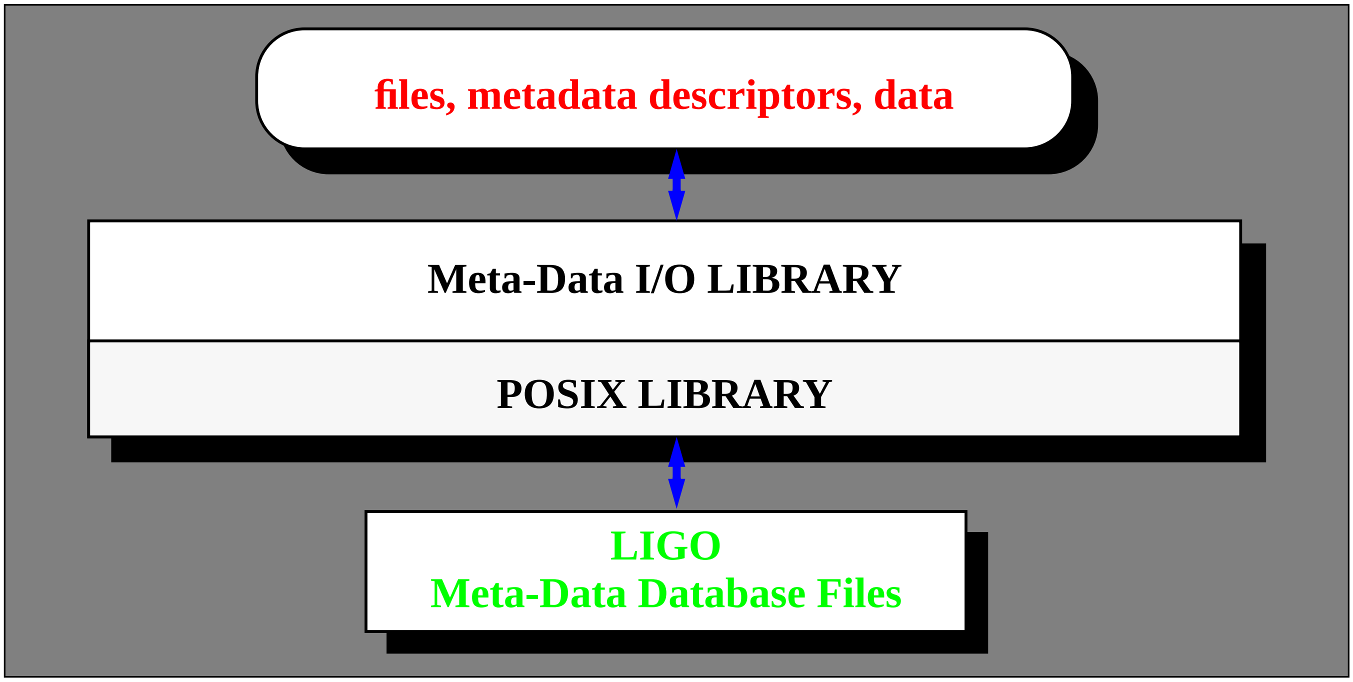
- Data:
  - Files, Light-Weight Descriptors, Data
  - Candidates: Net-CDF, SDF, ASCII
- Methods:
  - Read / Write Light-Weight Files
  - Add / Extract Light-Weight Descriptors
  - Add / Extract Data
  - Modify Light-Weight Components
  - Error Handling

Event Data I/O Library



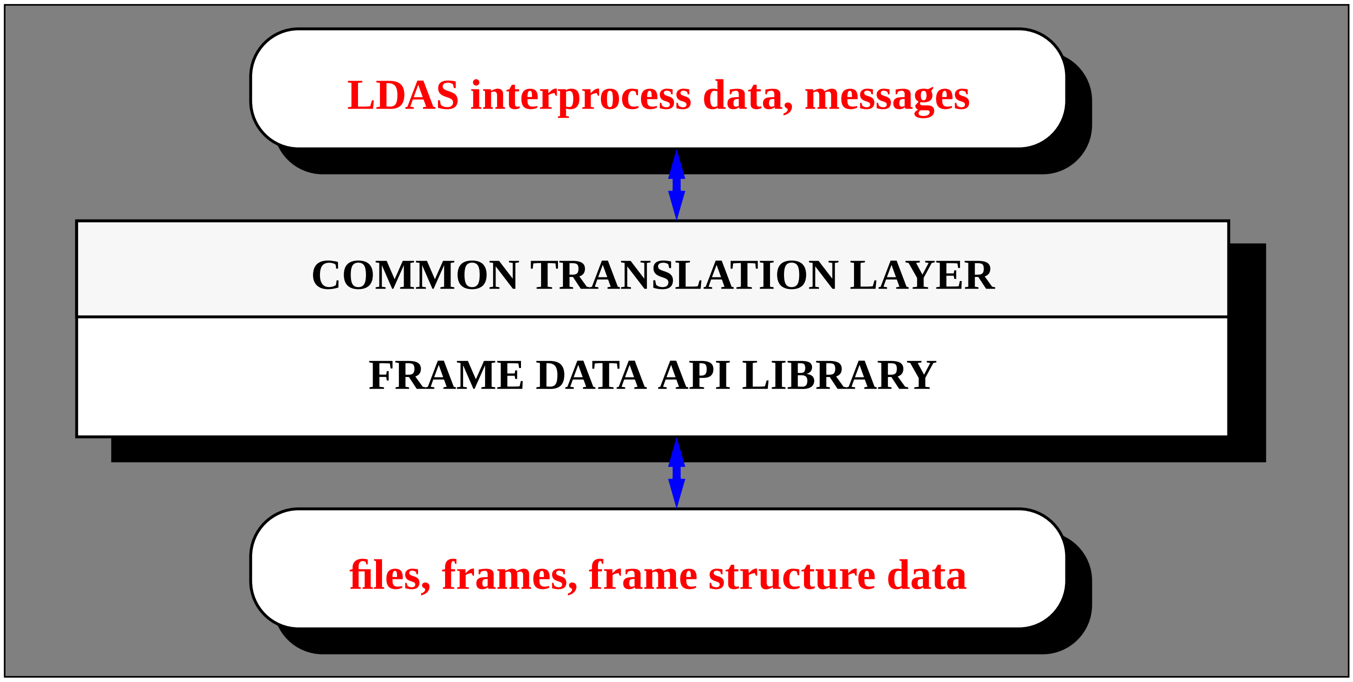
- Data:
  - Files, Event Descriptors, Data
  - Candidate: Database Technology
- Methods:
  - Read / Write Events
  - Add / Extract Event Descriptors
  - Add / Extract Event Data
  - Modify Event Components
  - Error Handling

Meta-Data I/O Library



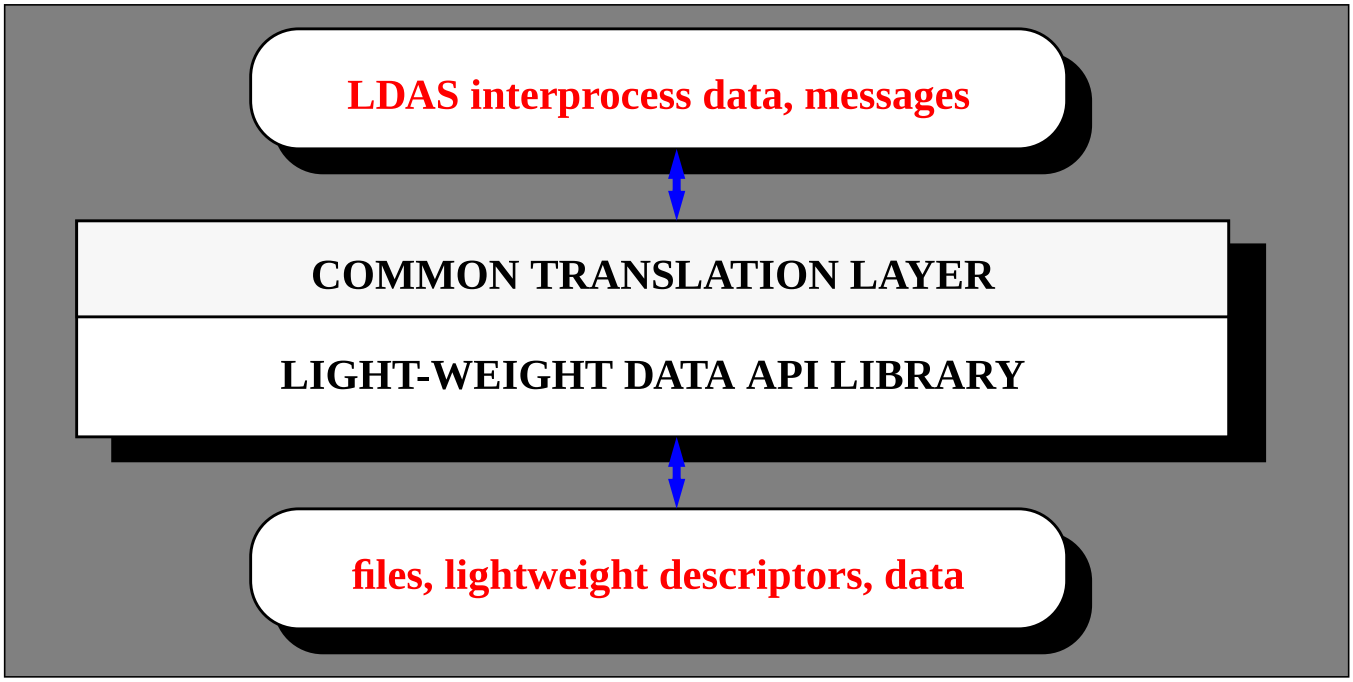
- Data:
  - Files, Meta-Data Descriptors, Data
  - Candidate: Database Technology
- Methods:
  - Read / Write Meta-Data
  - Add / Extract Meta-Data Descriptors (Fields)
  - Add / Extract Meta-Data
  - Modify Meta-Data Components
  - Error Handling

Frame Data API



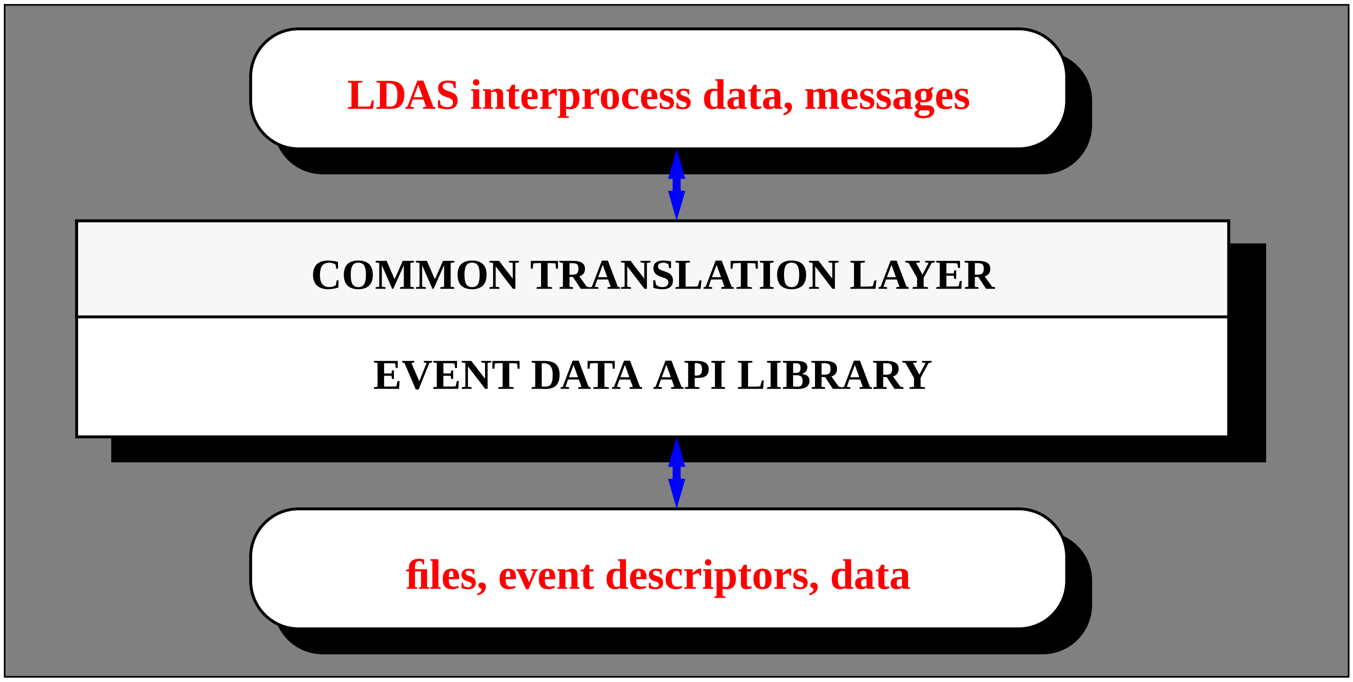
- Data:
  - LDAS Interprocess Data Formats
  - Files, Frames, Frame Structures, Frame Structure Descriptors, Data, Messages
- Methods:
  - Manipulate Data in Frame Format
  - Translate Data into LDAS Interprocess Data Formats
  - Receive / Process Messages from LDAS Manager API
  - Interface with Frame I/O Library
  - Error Handling

Light-Weight Data API



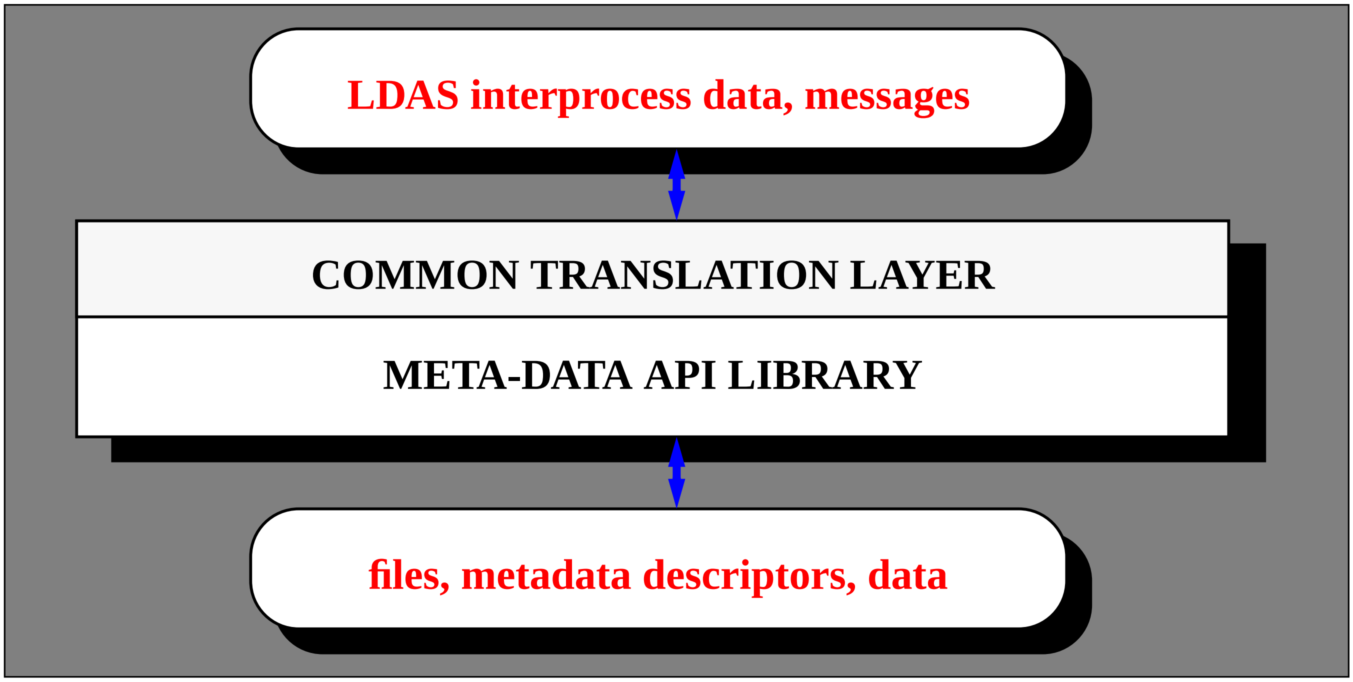
- Data:
  - » LDAS Interprocess Data Formats
  - » Files, Light-Weight Descriptors, Data, Messages
- Methods:
  - » Manipulate Data in Light-Weight Format
  - » Translate Data into LDAS Interprocess Data Formats
  - » Receive / Process Messages from LDAS Manager API
  - » Interface with Light-Weight I/O Library
  - » Error Handling

Event Data API



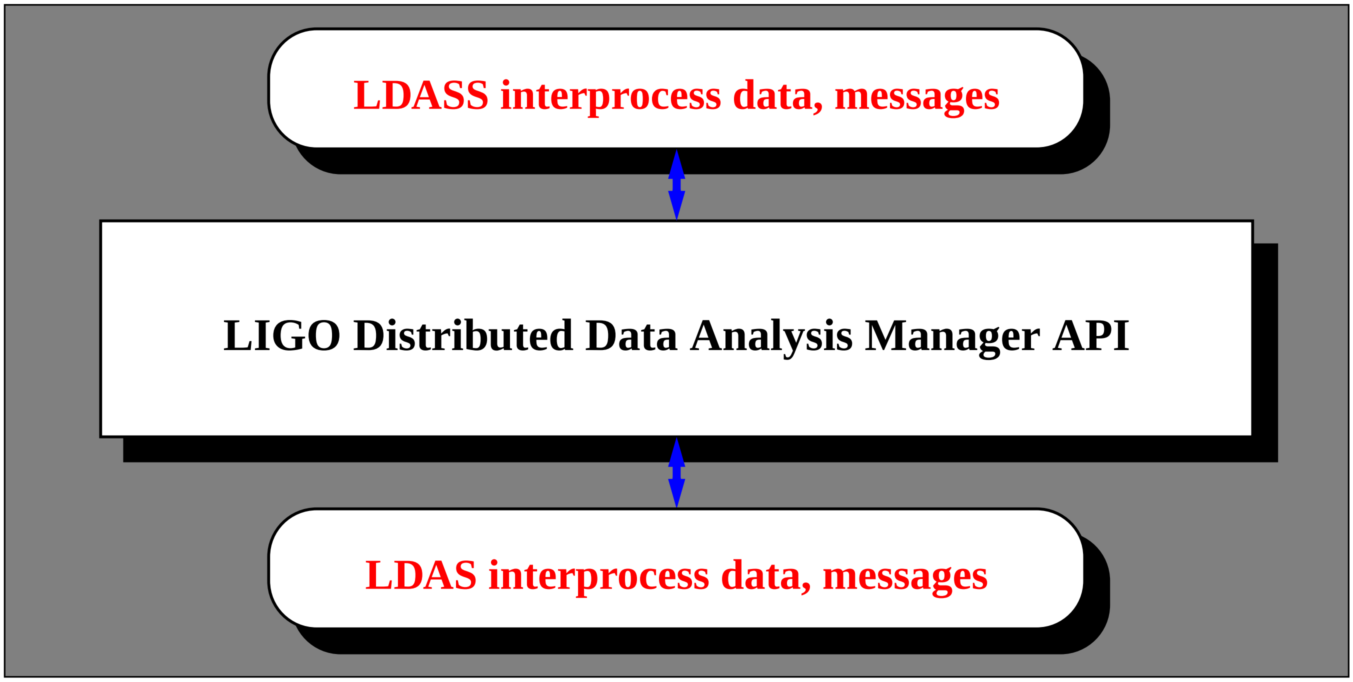
- Data:
  - LDAS Interprocess Data Formats
  - Files, Event Descriptors, Data, Messages(SQL)
- Methods:
  - Manipulate Event Data
  - Translate Data into LDAS Interprocess Data Formats
  - Receive /Process Messages from LDAS Manager API
  - Interface with Event Data I/O Library
  - Error Handling

Meta-Data API



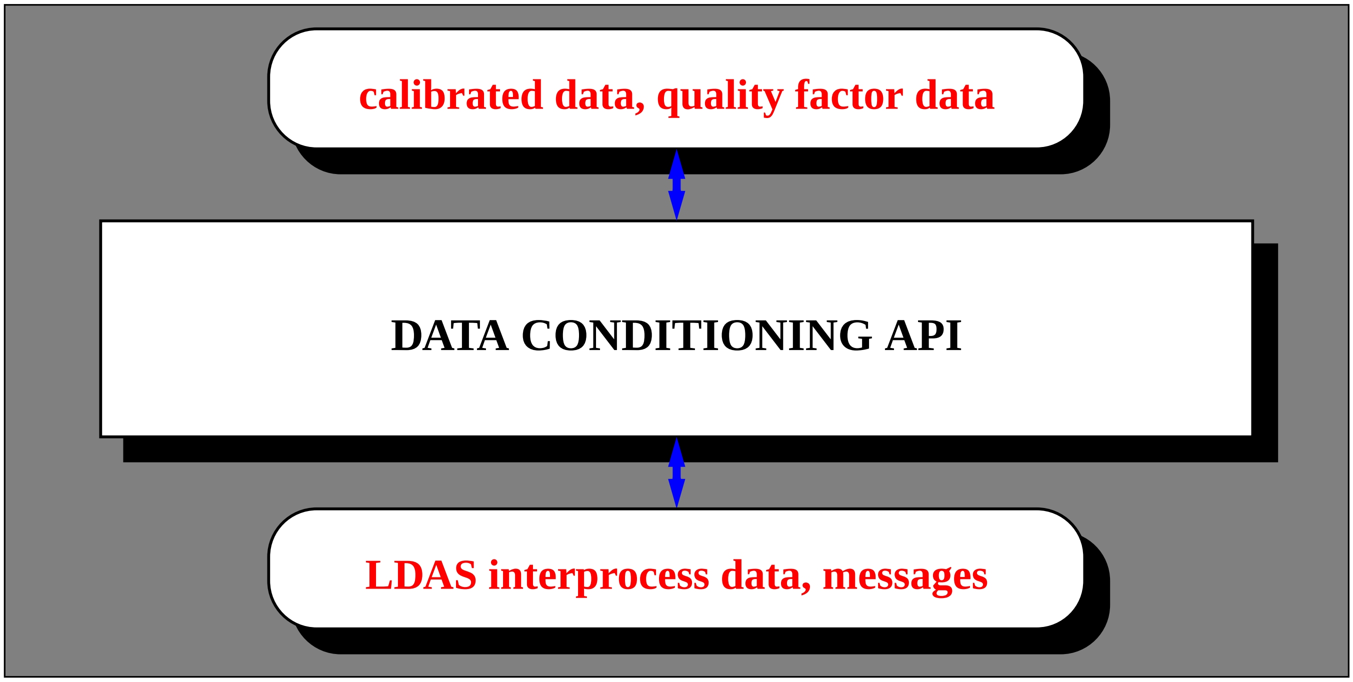
- Data:
  - LDAS Interprocess Data Formats
  - Files, Meta-Data Descriptors, Data, Messages(SQL)
- Methods:
  - Manipulate Meta-Data
  - Translate Meta-Data to LDAS Interprocess Data Formats
  - Receive / Process Messages from LDAS Manager API
  - Interface with Meta-Data I/O Library
  - Error Handling

Distributed Data Analysis Manager API



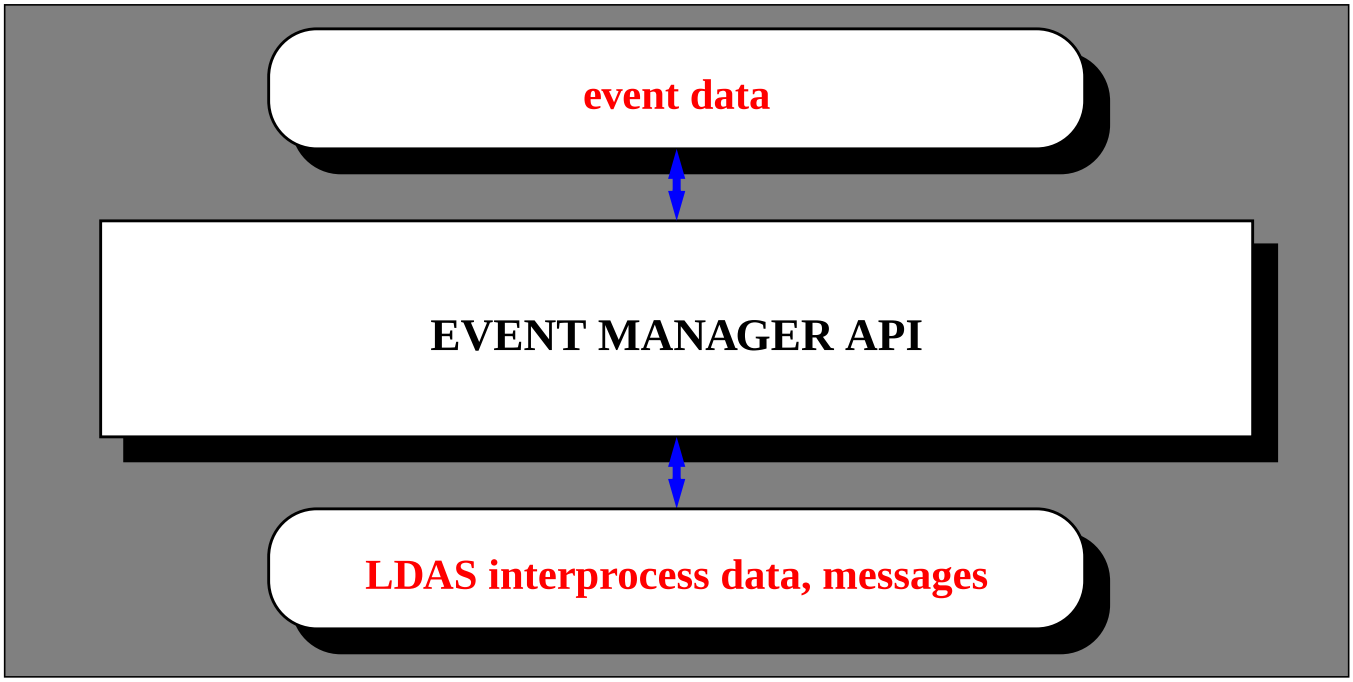
- Data:
  - LDAS Interprocess Data Formats, Messages
- Methods:
  - Manage LDAS Data Flow (& provide web services)
  - Send / Receive Messages
  - Distribute LDAS data among process
  - Establish TCP/IP Communications Links (including Web)
  - Error Handling

Data Conditioning API



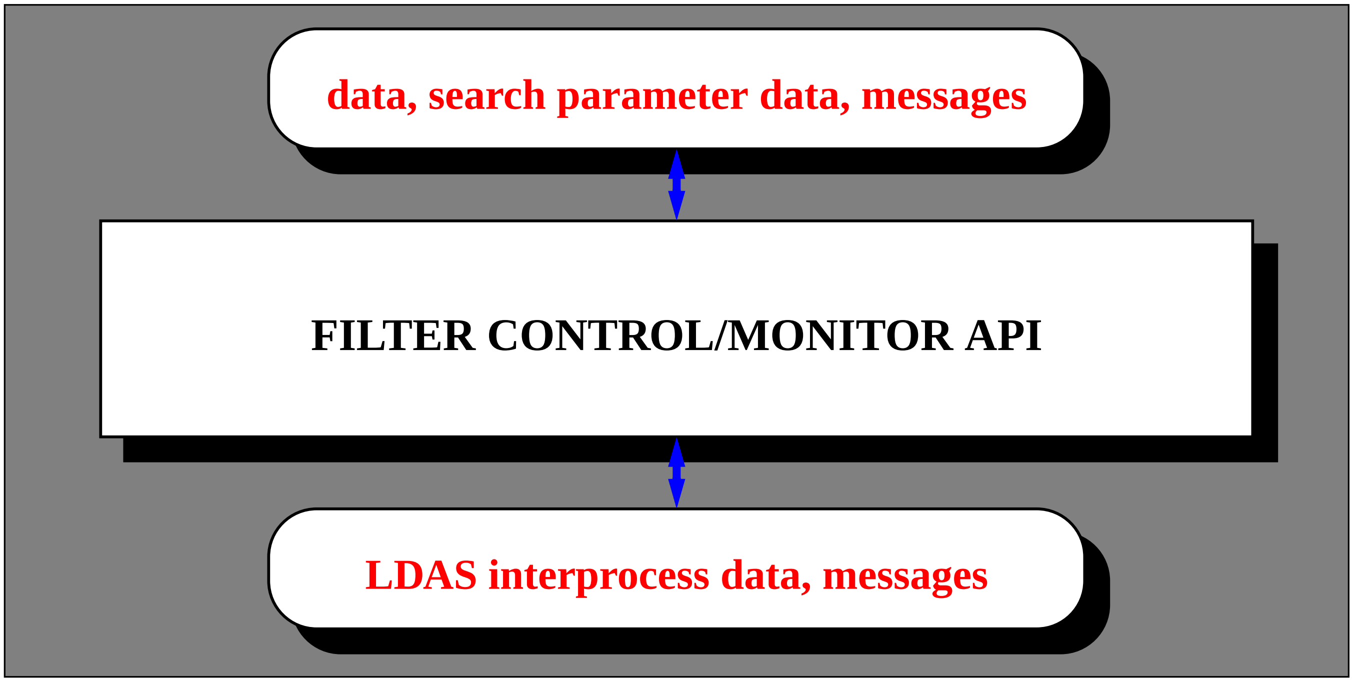
- Data:
  - Raw & Conditioned LDAS Data Formats, Messages
- Methods:
  - Send / Receive Messages
  - Condition (Calibrate, Regress, Etc.) Data
  - Produce Data Quality Factor
  - Distribute Conditioned Data with Quality Factor Data
  - Archive Conditioned Data
  - Error Handling

Event Manager API



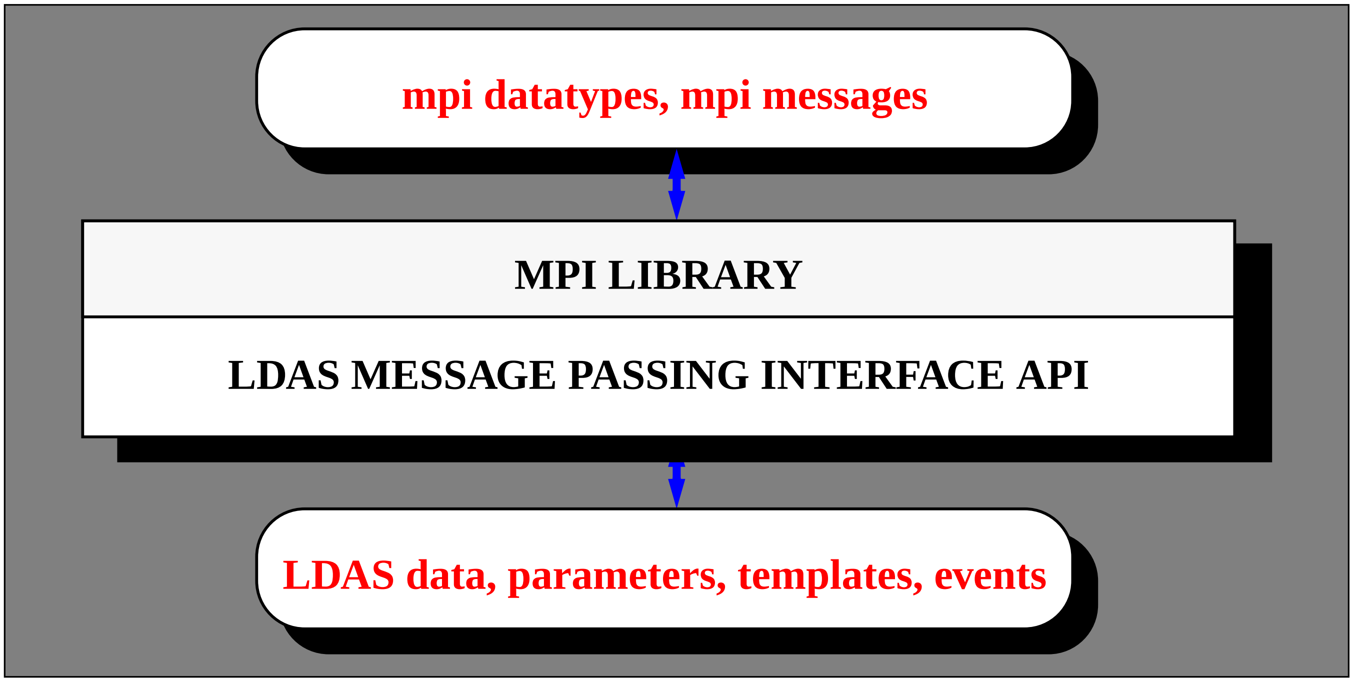
- Data:
  - LDAS Interprocess Data Formats, Messages
- Methods:
  - Send / Receive Messages
  - Stage Candidate Events in Hierachical Searches
  - Distribute Candidate Event Data
  - Archive Qualified Event Data
  - Error Handling

Filter Control/Monitor API



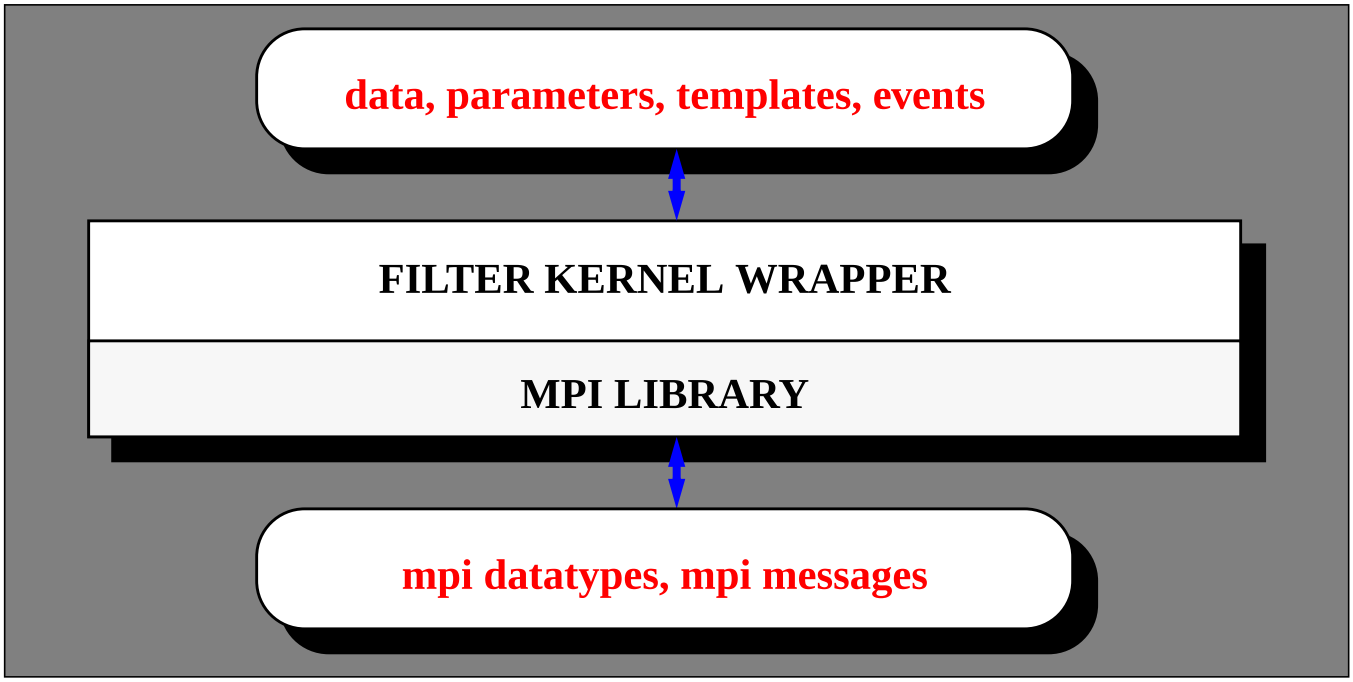
- Data:
  - LDAS Interprocess Data Formats, Messages
- Methods:
  - Provide Configureation Management Interface to LDAS
    - Stage Filters Used by All Searches
    - Distribute Parameters Used by Filters
  - Send / Receive Messages
  - Monitor Status and Performance of Filters and Data Flow
  - Error Handling

Message Passing Interface API



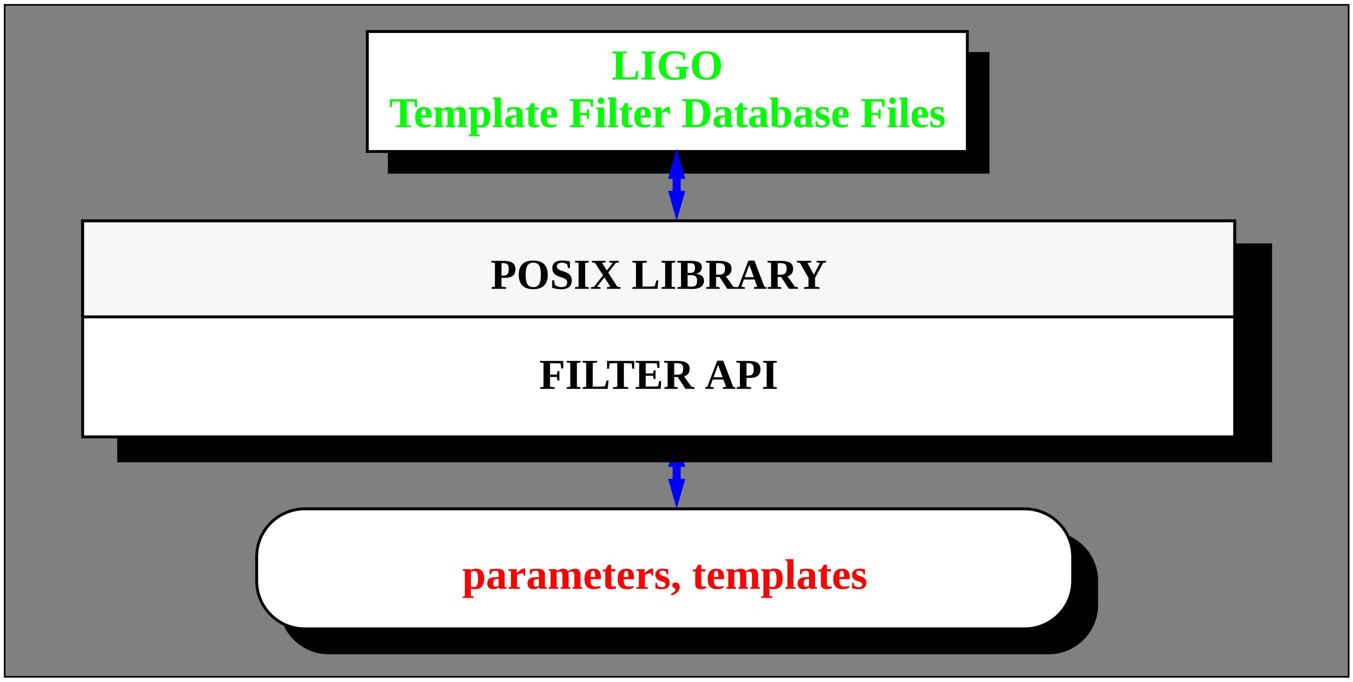
- Data:
  - LDAS Interprocess Data Formats, Messages, MPI Data, MPI Messages
- Methods:
  - Extend Send / Receive Messages of MPI-1 API
  - Distribute Conditioned Data and Parameters to Filters
  - Receive Filter Results (Candidate Event Data)
  - Control Filter Processing / Hardware Configuration
  - Error Handling

Filter Kernel Wrapper



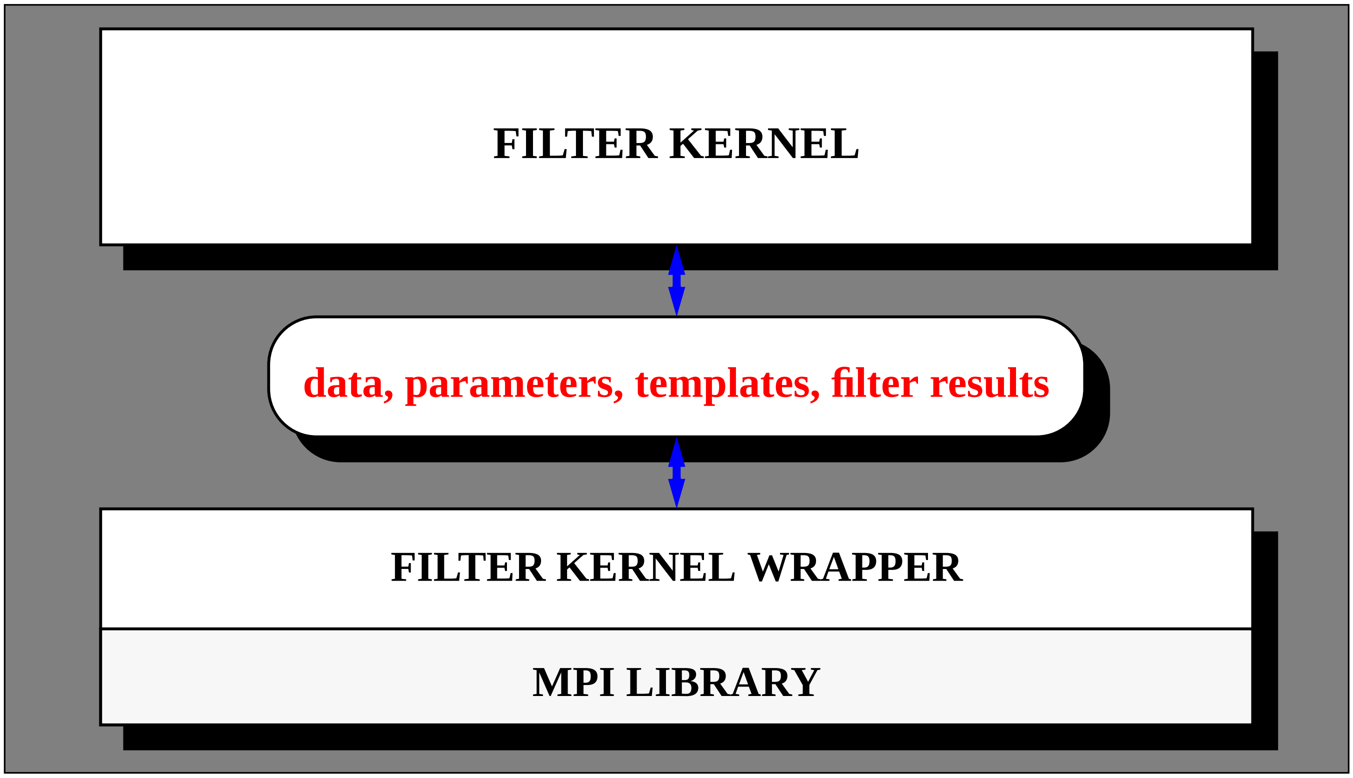
- Data:
  - LDAS Interprocess Data Formats, Parameters, Templates, Messages, MPI Data, MPI Messages
- Methods:
  - Abstract (Base Class) Methods for Individual Filters
  - Isolates filter algorithm implementation from LDAS fabric
  - Send / Receive Data from Filter and MPI APIs
  - Return Filter Results to Event Manager
  - Report Status to Control/Monitor API
  - Error Handling

Filter API



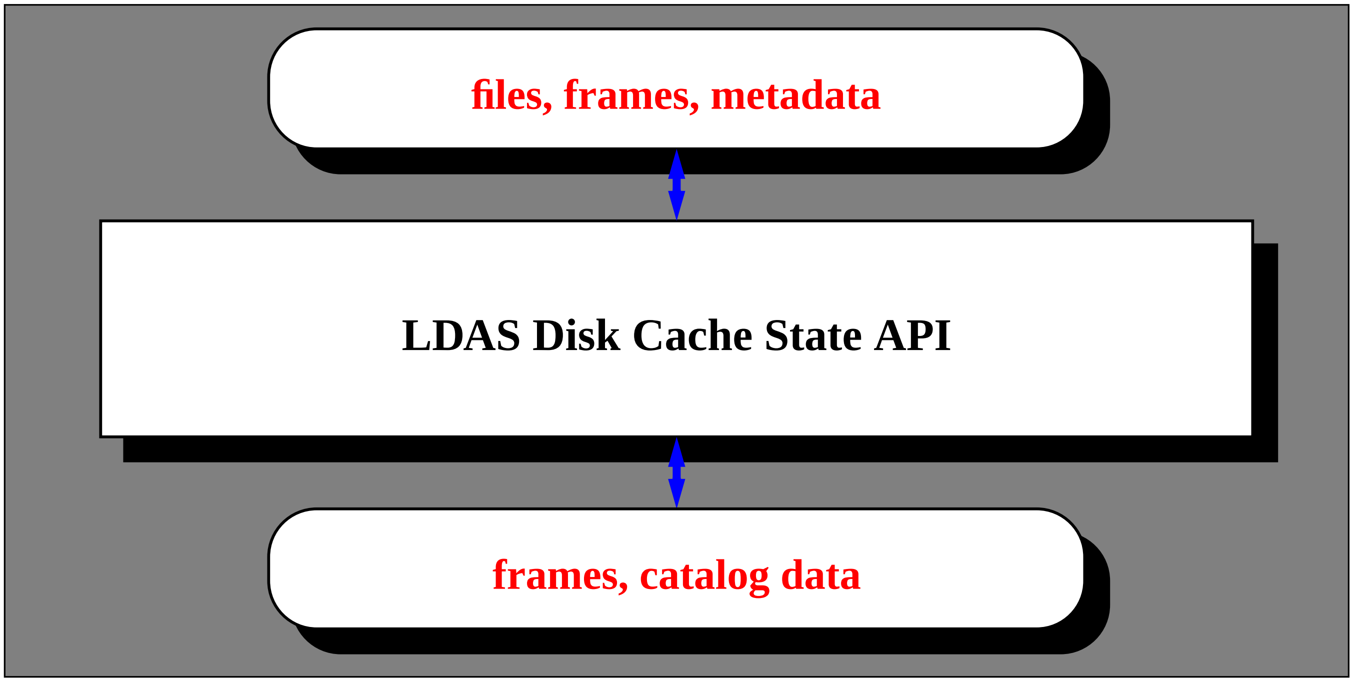
- Data:
  - Parameter Data, Template Data
- Methods:
  - Like NR but specialized for GW data analysis
  - Common signal analysis methods (FFT, etc.)
  - Calculate Filter Templates on Request
  - Read / Write Filter Templates onto File System
  - Return Requested Filter Templates to Filter Wrappers
  - Error Handling

Filter Kernel



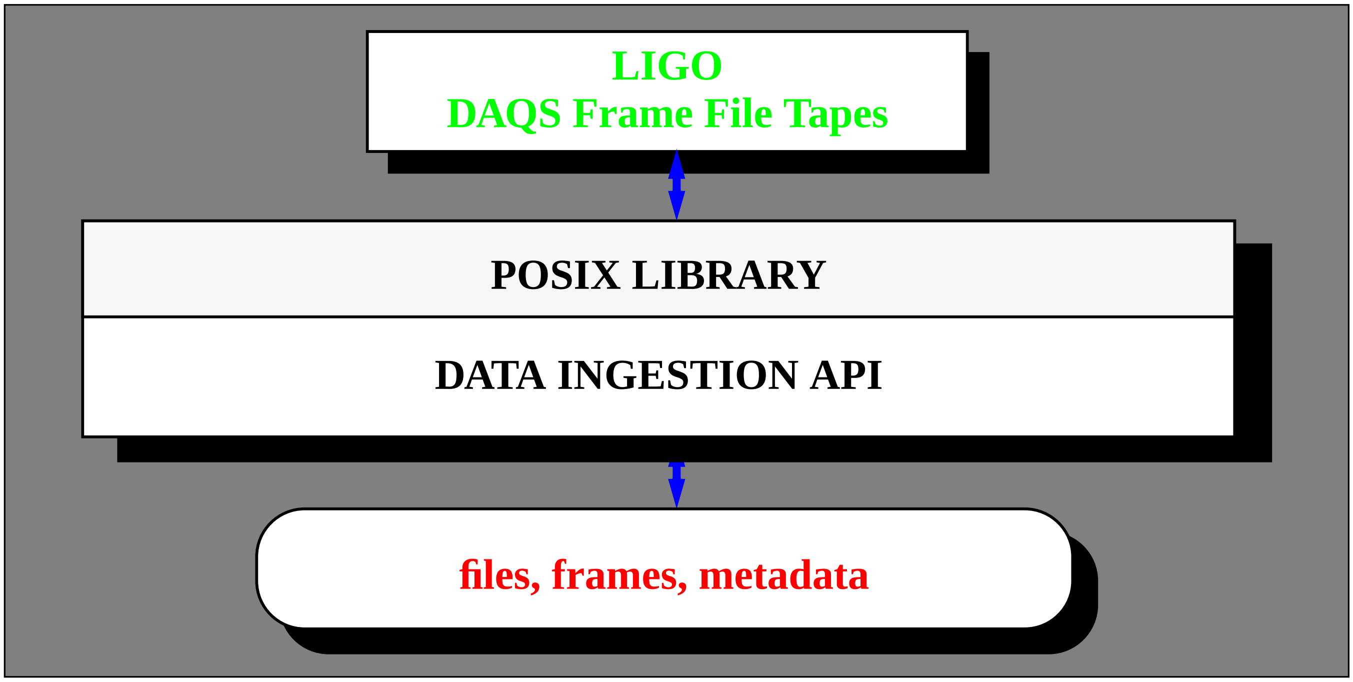
- Data:
  - Conditioned Data, Parameters, Filter Templates, Filter Results (Candidate Event Data)
- Methods:
  - LIGO researcher provided Algorithms
  - Analyze Data, Parameters, Templates
  - Report Back Results of Algorithm
  - Error Handling

LDAS Disk Cache State API



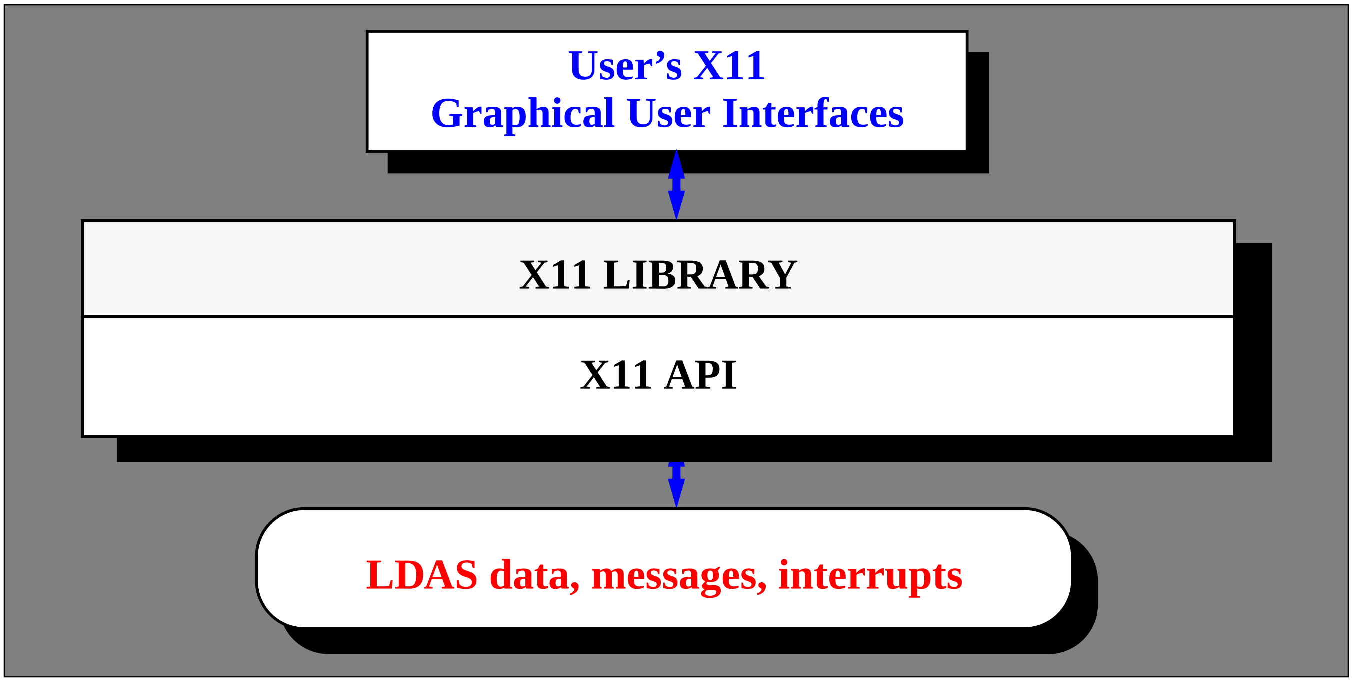
- Data:
  - LDAS Interprocess Data Formats, Messages
- Methods:
  - Add/Delete Frames(Files) from Disk Cache
  - Maintain Inventory of Disk Cache Contents
  - Distribute/Archive Frames found in Disk Cache
  - Error Handling

Data Ingestion API



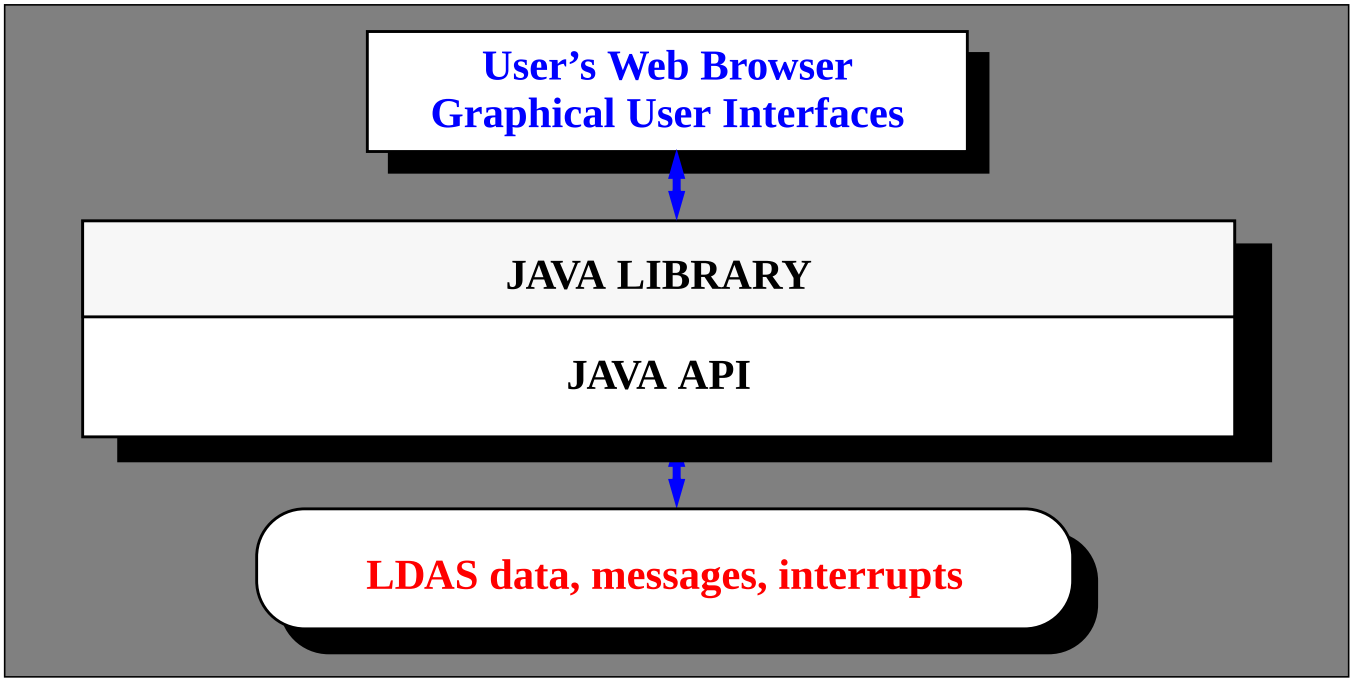
- Data:
  - Files, Frames, Meta-Data
- Methods:
  - Read Files/Frames stored on DAQS Tape
  - Produce Meta-Data about these Files/Frames
  - Report Read Files/Frames/Meta-Data to LDAS Disk Cache State API
  - Error Handling

X11 API



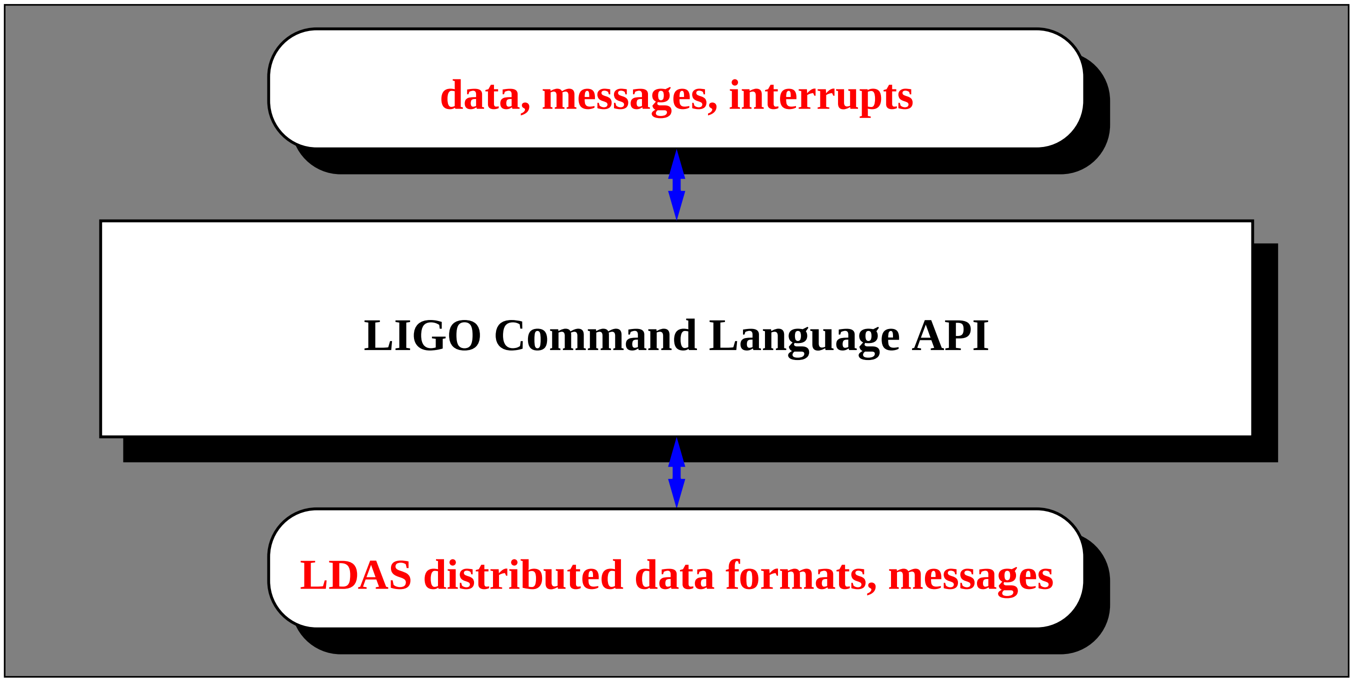
- Data:
  - User Provided Data, User Selections, Messages, User Generated Interrupts
- Methods:
  - Interpret User Generated GUI Events/Messages
  - Update/Maintain GUI windows on User Screens
  - Parse User Messages into LDAS Command Language
  - Error Handling

Java API



- Data:
  - User Provided Data, User Selections, Messages, User Generated Interrupts
- Methods:
  - Web Browser Interface
  - Interpret User Generated GUI Events/Messages
  - Update/Maintain GUI windows on User Screens
  - Parse User Messages into LDAS Command Language
  - Error Handling

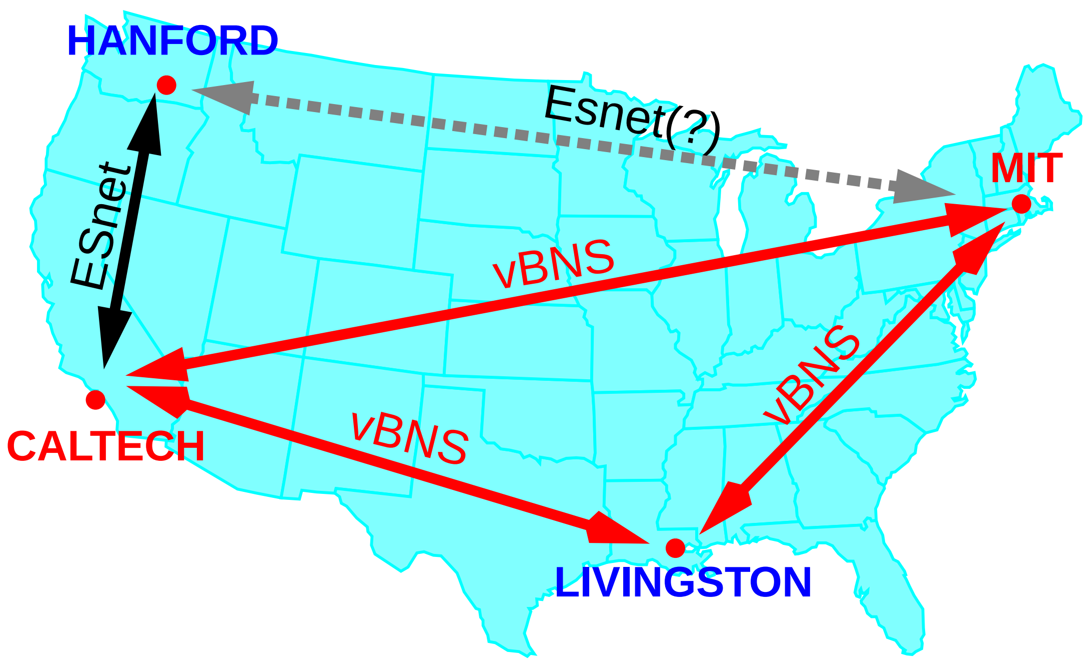
LDAS Command Language API



- Data:
  - LDAS Distributed Data Formats, Messages
- Methods:
  - Provide Shell Scripting Language Interface to LDAS
  - Send / Receive Messages and Data
  - Interface with GUI APIs
  - Establish TCP/IP Communications Links (including Web)
  - Error Handling

NETWORKING

LIGO Wide Area Network



- LIGO drafted proposed MOU between NSF/DOE to establish access to ESnet at Hanford -- presently @ NSF for review.
- Working with LSU to set up link to Livingston; access to vBNS when available @ LSU

WAN/LAN Connectivity among LIGO Laboratory Sites

Site	*Livingston, LA*	*Hanford, WA*	*MIT*	*Caltech*
Caltech	**vBNS/OC3**	**ESnet (3 X T1) <-> vBNS/OC3**	**vBNS/OC3**	**OC3/ATM 100BT**
MIT	**vBNS/OC3**	**ESnet (3 X T1) <-> vBNS/OC3**	**100BT OC3(?)**	
Hanford, WA	**ESnet (3 X T1) <-> vBNS/OC3**	**OC3 100BT**		
Livingston, LA	**OC3 100BT**			

LIGO is an Active Library

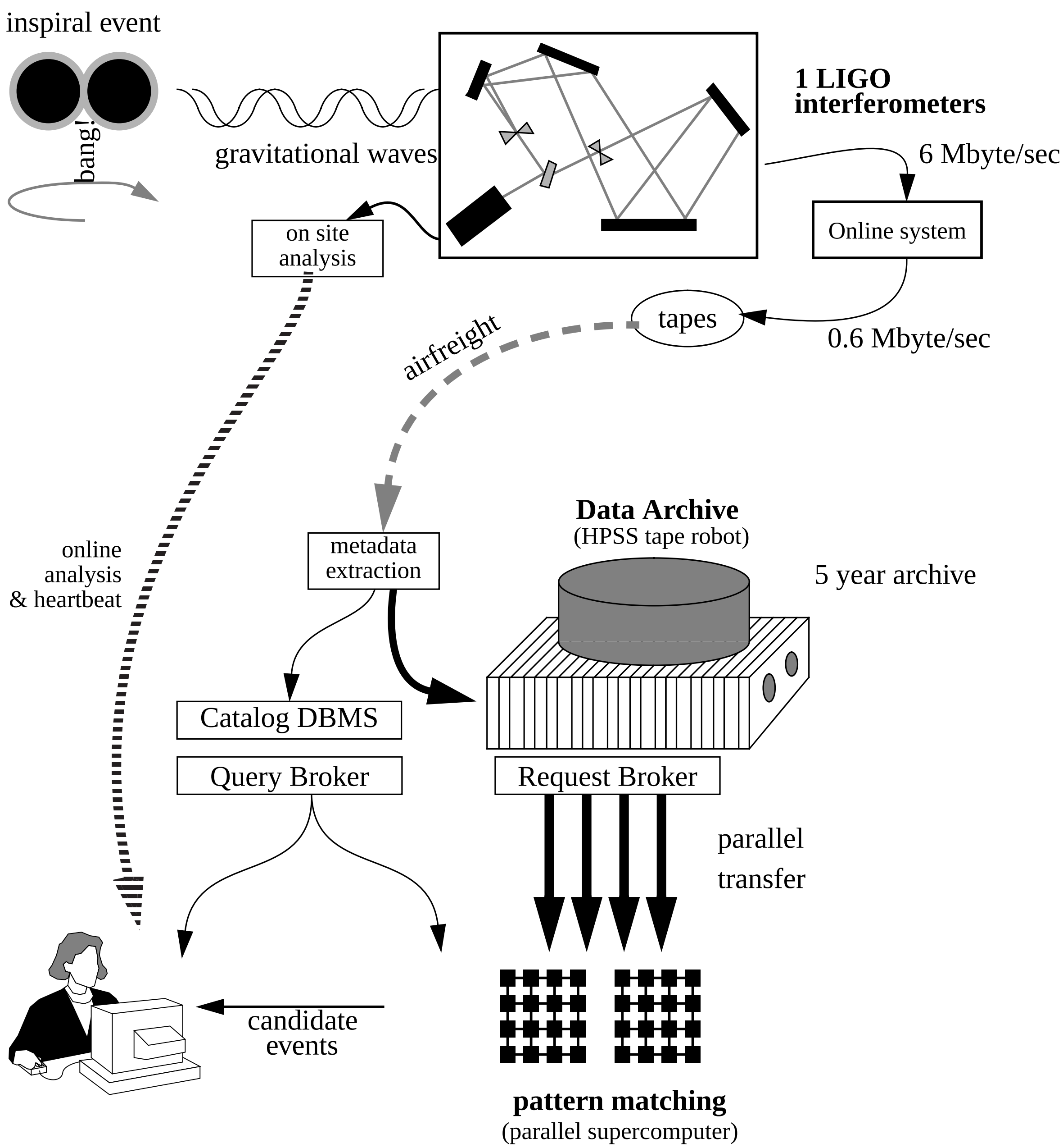
(User Services from LDAS)

Roy Williams

Center for Advanced Computing Research
California Institute of Technology

(and its Flexible,
with up-to-date technology, and
each program runs on each platform, using
modular, extensible software)

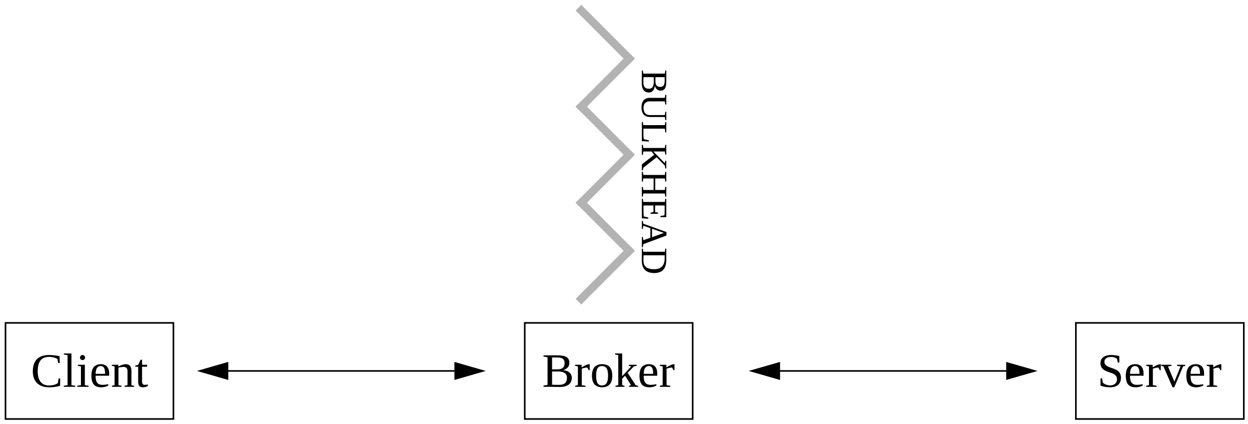
LIGO Dataflow



Schematic data flow for LIGO. The interferometers provide a real-time stream to on-line analysis, whose results are broadcasted to interested users. The bulk of the data is written to tape and airfreighted to the CACR, where it is catalogued and archived. Supercomputers use the catalog to convert a high-level query to simple file requests, which are satisfied by a request broker attached to the archive. Parallel data streams flow to the pattern matching code and candidate events are sent back to the user or put into another database.

The Three-Tier Model

Reduces risk
Reduces development time
Simplifies maintenance



Published Interface
Library API

Filtering
Translation

File archive
Databases

Query language
- Locked data
- Data from last week
- List of gamma-ray bursts
- Candidate events with SNR > 6

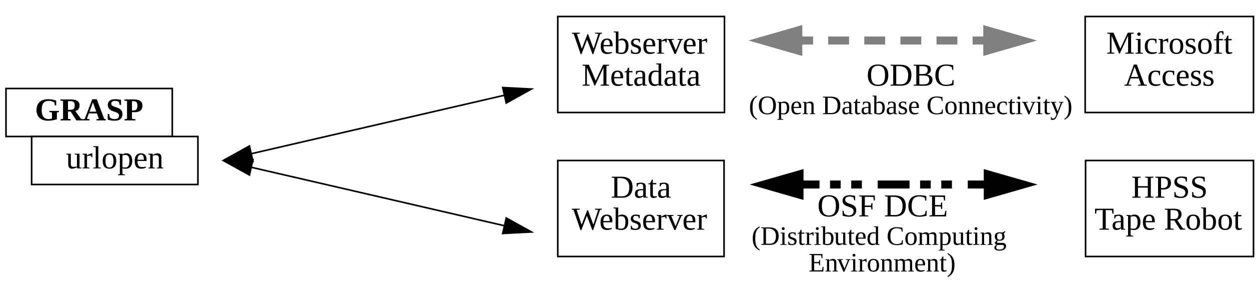
File formats
- Frames
- netCDF
- PAW tuples
- Audio
- Lists of file names

Implementation
- Finding the data
- Mounting tapes
- Scheduling
- Objectivity
- MS Access
- HPSS
- Cybernetics
- ATM
- DCE/DFS

Examples of the Three-Tier Model

Note hiding of complex protocols
Note that Microsoft and/or HPSS can be transparently replaced

Client fills in a form on a web browser asking what data is available between two dates.



Production code is linked to GRASP and the urlopen API.
Metadata server provides a list of filenames, and each is used to generate a query to the data server.

Examples of the Three-Tier Model

Datawolf Intelligent Archive

Query Example

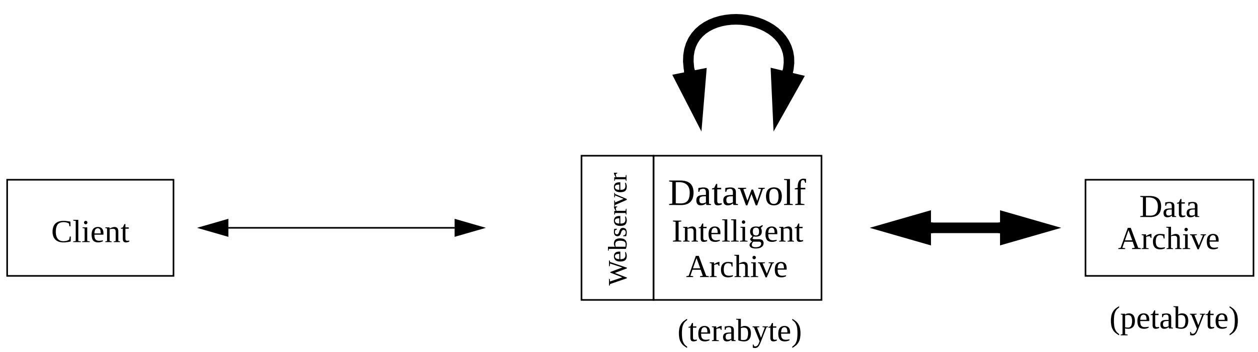
I want all the data from the working set where Channel 67 is greater than 4

Working set:

A small subset of the archive on spinning media

Intelligence:

Processing power connected at high bandwidth to the Working Set



Client ⟷ Broker ⟷ Server

Queries, not Files!

Think of queries, not files, formats, databases or locations

Query input

Keyword-value, for example in input to the frame catalogue
T0=76083737
T1=76086574

SQL-like
SELECT Channels.IFO, Channels.Lock AS Expr FROM Channels
WHERE ((Channels.Lock.Value>55) AND (Time > T0) AND (Time < T1));

Query Output (If it is to be read by a computer)

Frame file names

Other mime-typed files
eg. application/x-netcdf, image/jpeg, text/plain, ...

Table
eg Gamma-ray Bursts...

Date	SNR	RA	Dec

Report
derived from a Table

Some Servers

Frame files

Give me a file (equivalent to ftp)
FILE=14nov94.1/94-11-14-10-21-54

Give me the data and format that I want
CHANNEL=seismometer
FORMAT=SDF

Intelligent requests

Give me certain channels between Tstart and Tend
But only when conditions are satisfied
When there is a long contiguous chunk
In the format that I want

Metadata about the frames (the "Frame Catalogue")

When is the instrument working?
Tstart=76083737,Tend=76086574
produces a list of time intervals
76083737 76083739
76083743 76083748
76083760 76083769

Given a frame file, what are the names of the channels in it?

External events

Such as earthquakes, gamma-ray bursts
Each event has a time, a significance, and "other data", with table output
76083737 3.78 blah blah
76086458 2.33 blah blah

Candidate events from Ligo data,

Given a time interval T0,T1 what is in the collection
From which template set?
Authority information such as authors, likelihood, processing documentation

Instrument Calibrations and Settings

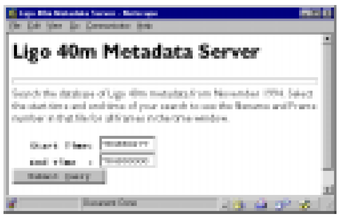
Hierarchical
Historical.
Collections of comments logged by operators

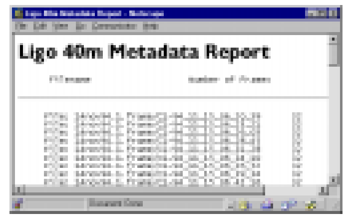
Constructing a Query: Some Metadata

http://sara.cacr.caltech.edu/index.htm

Finding out file names for 40-meter data between two times

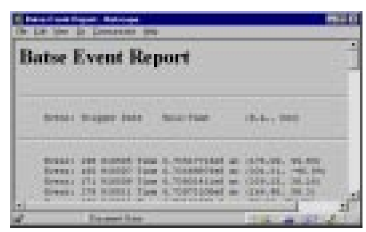
file-descriptor = **urlopen("**
http://sara.cacr.caltech.edu/ligo/getframes.htf?
Tstart=784880277&Tend=784900000**");**





Gamma-ray bursts between two times

file-descriptor = **urlopen("**
http://sara.cacr.caltech.edu/ligo/getbatse.htf?
Tstart=673500000&Tend=683500000**");**



Constructing a Query: Getting Real Data

http://www.cacr.caltech.edu/ligo
login: ligo
password: <will be announced>

---

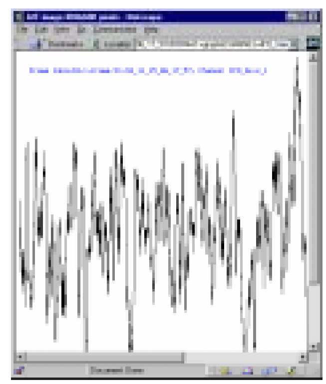
file-descriptor = **urlopen(“**
http://bitty.cacr.caltech.edu:1080/cgi-bin/ligo_frame_server?
FILE=14nov94.1.frame/C1-94_11_15_06_17_57&FORMAT=graph&CHANNEL=IFO_Seis_1**”);**

FILE=name

FORMAT=text
FORMAT=audio
FORMAT=SDF
FORMAT=NetCDF
FORMAT=graph
FORMAT=raw

CHANNEL=IFO
CHANNEL=Arm_Coil
CHANNEL=Microphone
etc etc

Graphs, Audio, Text dump, etc



Getting Frame files

FORMAT=raw

Can take the file descriptor and give it to
FrOpen_Fd (open Frame file from file descriptor)

Time Representation

Human time representation: **10/28/97 at 2:12:17 pm PST**

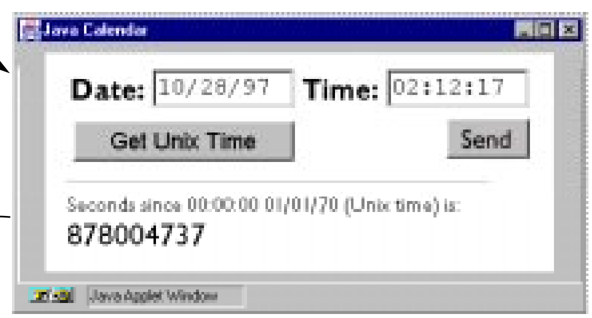
Easy transformation

Difficult and Error-prone

Unambiguous time representation: **878004737**
(Seconds since epoch)

We can push the input data through independent functions or applets to massage it to the correct form

Human time
10/28/97 at 2:12:17 pm PST



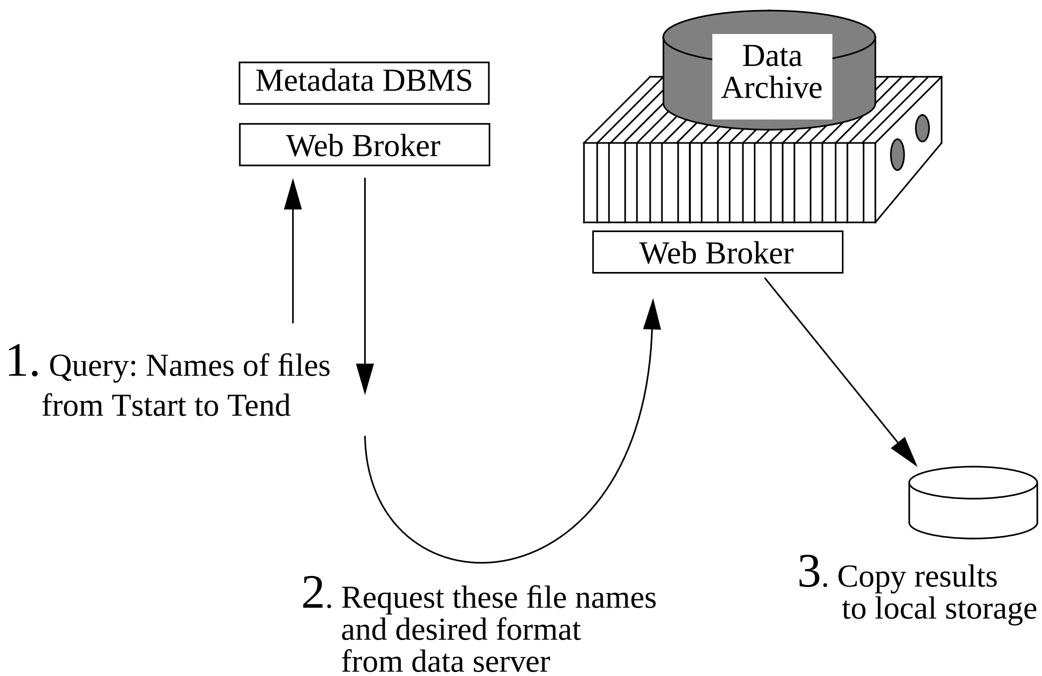
http://sara.cacr.caltech.edu/ligo/getframes.htf?
Tstart=878000000&Tend=878004737

Parastage: Staging Data from Archive

parastage [flags] querystring

Stages data to **local storage** from a web-based archive, meaning two servers, a **data server** and a **metadata server**.

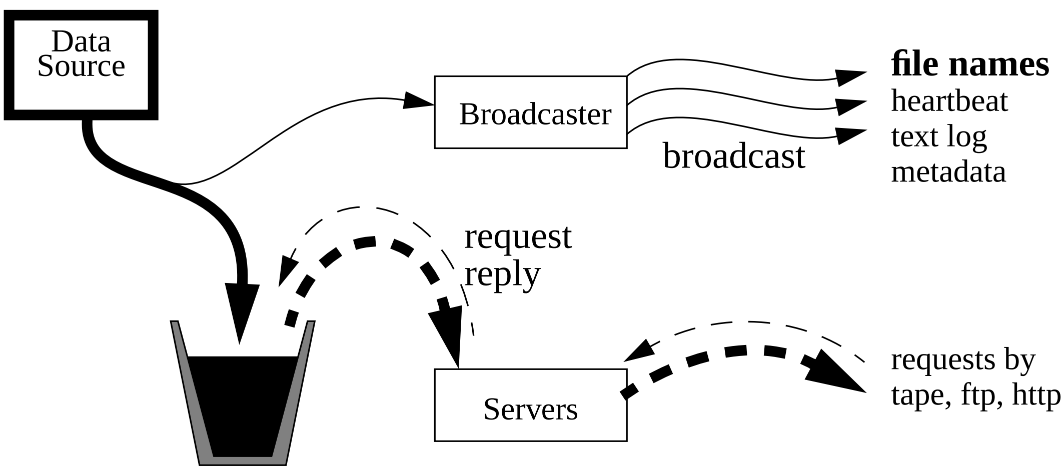
These servers are **brokers**.
The interface is a web server, but the implementation can change .



Channels: Let’s get Real-Time

Diagnostics
Monitoring health
Remote on-line processing
Visualization displays
Sensitivity analysis

Online
Real-time data
Brownout means dropped data
Server pushes to client



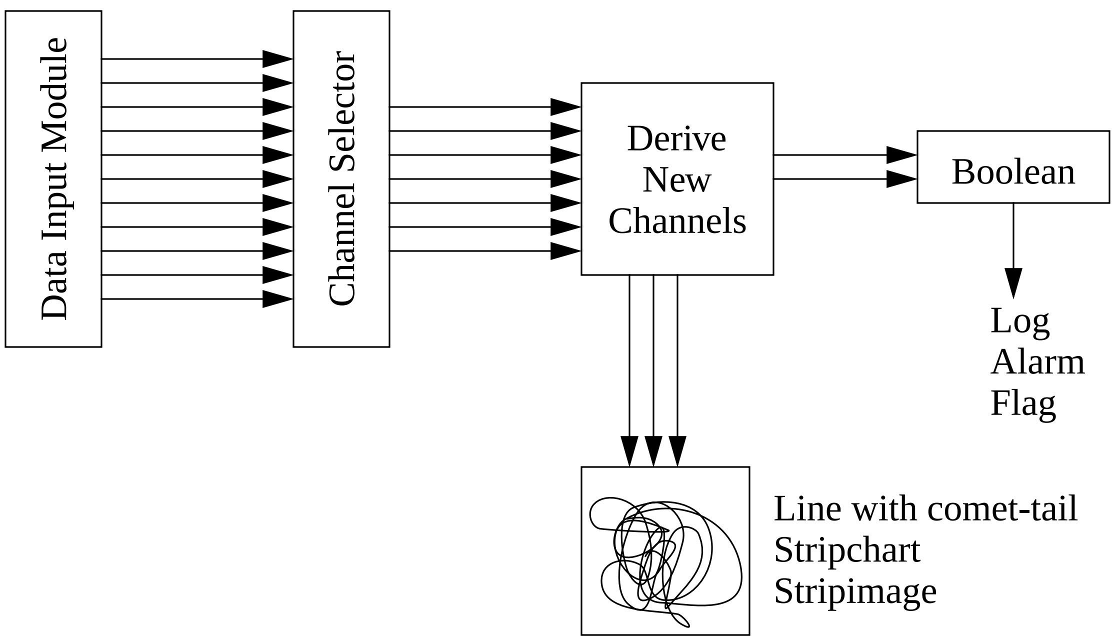
Offline
Time is not counted
Brownout means slow, reliable data
Client requests from Server

A Workspace for Multichannel Time-Series

Multichannel time series

$$f_i(t) \qquad i = 1,\ldots, N \quad t \text{ is real}$$

Assume slow timestep is a multiple of fast channel
We can, if necessary, interpolate arbitrarily to real time



New channels could be

Principle components
Components with respect to a moving average of the principle components
Fourier coefficients
Components with respect to a template set

Question

What conceptual changes are needed to extend to frequency space?

LDAS PROTOTYPING

Data Distribution and Testing

- Current of Frame I/O Library supports data compression:

GZIP Level	Differentiation	Translated Frame Size	Frame Size vs. Raw Data Size	Time (cpu) to Translate
None	No	1282532 KB	97.67%	975s (7.4%)
1	Yes	667693 KB	50.85%	1461s (75%)
1	No	726269 KB	55.31%	1494s (72%)
3	Yes	640549 KB	48.78%	1799s (78%)
3	No	706373 KB	53.80%	1863s (77%)
6	Yes	621157 KB	47.31%	3951s (91%)
6	No	697533 KB	53.12%	3187s (83%)
9	Yes	619965 KB	47.21%	4940s (91%)
9	No	696613 KB	53.05%	4401s (87%)

›› Translation benchmarks for 16483 seconds of Nov 94 40 meter data:

– table gives results for 200 MHz, single CPU Sun Ultra2 workstation,
(300MHz Ultra 30 gives 1.5X to 3X improvement in performance times!
e.g., GZIP = 1; Differentiation = Yes; became 798 seconds @ 97.4% CPU)

– nearly 50% reduction using 10x CPU increase & 1.5x increase clock time

– compression provides for direct savings in media!

Data Distribution and Testing

- Implemented Client-Server Frame I/O support to DAQS
  - Design uses TCP/IP “connected” Unix Sockets (portable & flexible)
- Tunable parameters allow optimal performance
- Four hardware/OS configurations tested
  - Between Sun workstations via 10baseT ethernet
    - Peak performance 1.1 MB/sec
    - 60 MB/sec between two processes on same Sun Ultra workstation
  - Baja MIPS processor to Sun Sparc10 via 10baseT ethernet
    - Peak performance 1.1 MB/sec
  - Baja MIPS processor to Sun Ultra via 100baseT ethernet
    - Peak performance 5.9 MB/sec (satisfies LIGO bandwidth needs!)
- C++ class communication library using sockets planned.

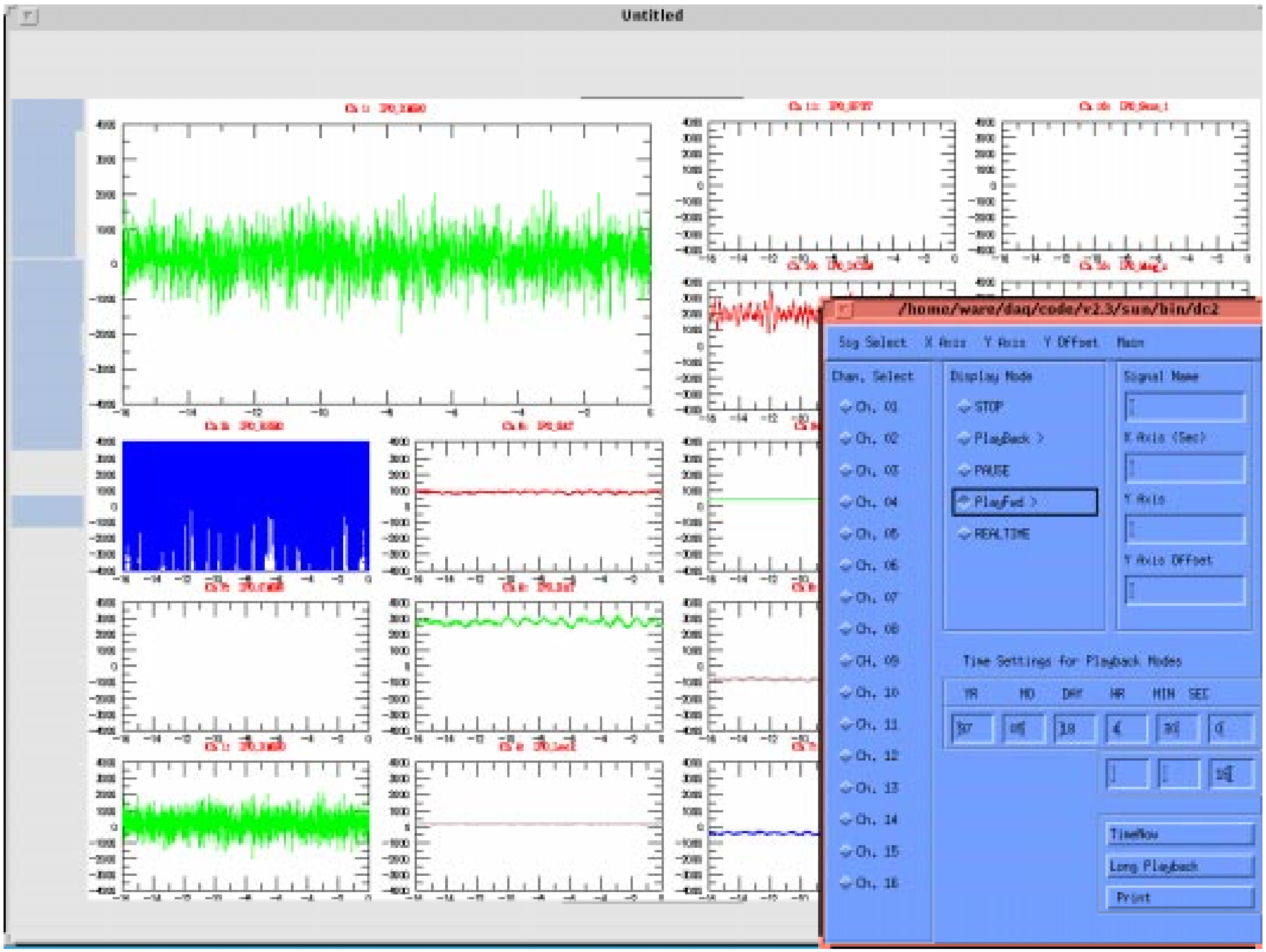
Data Distribution and Testing

- High Speed Network Tests at CACR
  - ›› 100GB on Sun transferred over ATM to HPSS at bandwidth of 4.4MB/sec
- Bandwidth between protocols tested

	HIPPI (FP) Paragon-HPSS server	HIPPI (TCP/IP)	ATM (TCP/IP)
HIPPI (FP) Paragon-HPSS server	Mem-Mem: 50MB/sec Parallel Filesystem-Mem: 30MB/sec	-	-
HIPPI (TCP/IP)	-	HPSS Server-HPSS Server: 6MB/sec Paragon-HPSS Server: 2.4MB/sec	Sun-HPSS Server: 1MB/sec HPSS Server-Sun: 0.2MB/sec Sun-SP2: 2MB/sec
ATM (TCP/IP)	-	Sun-HPSS Server: 1MB/sec HPSS Server-Sun: 0.2MB/sec Sun-SP2: 2MB/sec	Sun-Sun: 9-14MB/sec SP2-SP2: 10-15MB/sec Sun-HPSS Server: 7-9MB/sec

- ATM - ATM (TCP-IP) ~10MB/sec close to adequate for two full IFO data streams

Quick-Look Analysis Tools



FUTURE LDAS PROTOTYPING

- Software:
  - C++ Class Library for the Common Data Format (FRAMES)
  - C++ Class Library for socket based interprocess communications
  - C++ Class Library evolution for relavent GRASP components
  - Continued definition of C++ Classes for LDAS Software Block Diagram
  - Configure LDAS CVS client-server mode version control system
  - Continue prototyping Paraflow/Parastage model
  - Continue analysis and evaluation of tools for quick-look of 40 meter data
- Hardware:
  - Establish ATM link between 40 meter lab and CACR
  - Procure small BEOWULF system to become familiar with this technology and begin development of LDAS software components on this platform
  - Add Data Conditioning/Data Distribution Compute Server with multi-host mass storage drives to 40 meter lab
  - Test CACR HPSS (High Performance Storage System) as 40m lab data archive
  - Begin testing WAN performance as ESNet & vBNS become available at sites and universities

LDAS Development
Schedule

ID	Task Name
1	***White paper released***
2	**DAS Design Requirements**
3	**Hardware architecture**
4	Requirements Definition
5	Conceptual Design
6	**Software design**
7	Software Standards
8	Software Func. Requirements
9	Network architecture
10	DRR draft distribution
11	DAS DRR
12	**DAS Preliminary Design**
13	Preliminaray Design Document
14	PDR Draft Distribution
15	DAS PDR
16	**DAS Final Design**
20	DAS FDR
21	Hardware Procurement
22	Hardware install & debug
23	Hardware prototyping activities
24	
25	Software design
26	**Software development**
27	**Data handling software**
28	C++ Frame library
29	Networking software implementation
30	Storage system implementation
31	Database/archiving implementation
32	Dataflow implementation
33	Software integration
34	Software validation
35	**Software prototyping**
36	GUI & Paraflow
37	Theoretical hardware benchmarks
38	Actual benchmarks
39	**Algorithm development**
40	Algorithm Requirements Definition
41	Binary Inspiral
42	Black Hole Ring-down
43	Periodic Sources (pulsars)
44	Stochastic Background
45	Transients
46	Signal Conditioning
47	Time-Frequency Methods
48	Algorithm Reference Manual
49	**WA CDS DAQ Ready**
50	**LA CDS DAQ Ready**
51	**Begin LIGO Detector Oeprations**