LIGO Data Analysis System

LIGO PAC 7

Massachusetts Institute of Technology

November 18th - 19th, 1999

James Kent Blackburn
LIGO Laboratory
California Institute of Technology

LIGO-G990114-00-E

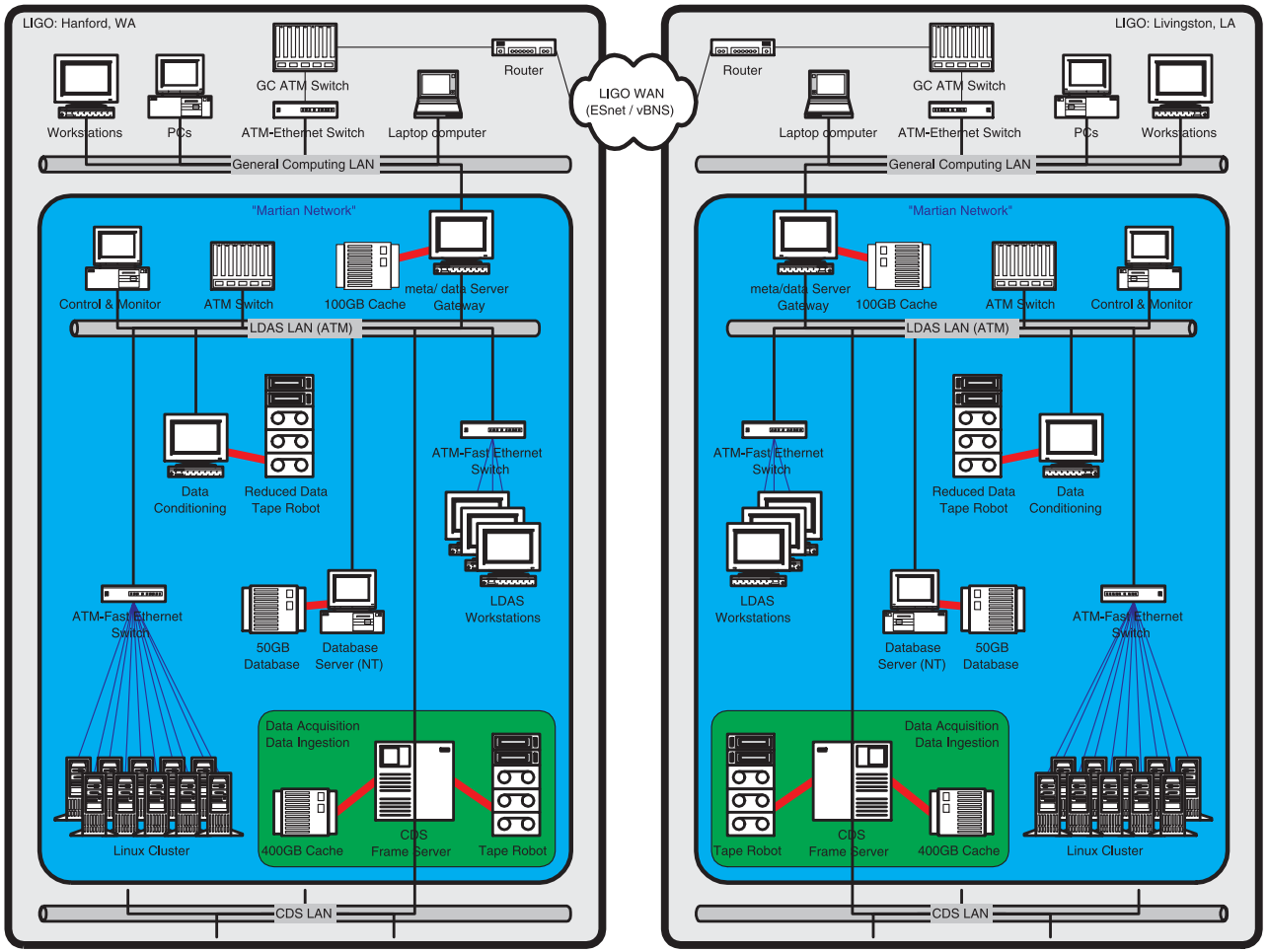
Introduction

- System Design Overview
  - *if requested*
- Site Installation
- Software Construction Review
- Software Documentation
- System Testing
- Next Major Development (Parallel LDAS)
- Closing Remarks

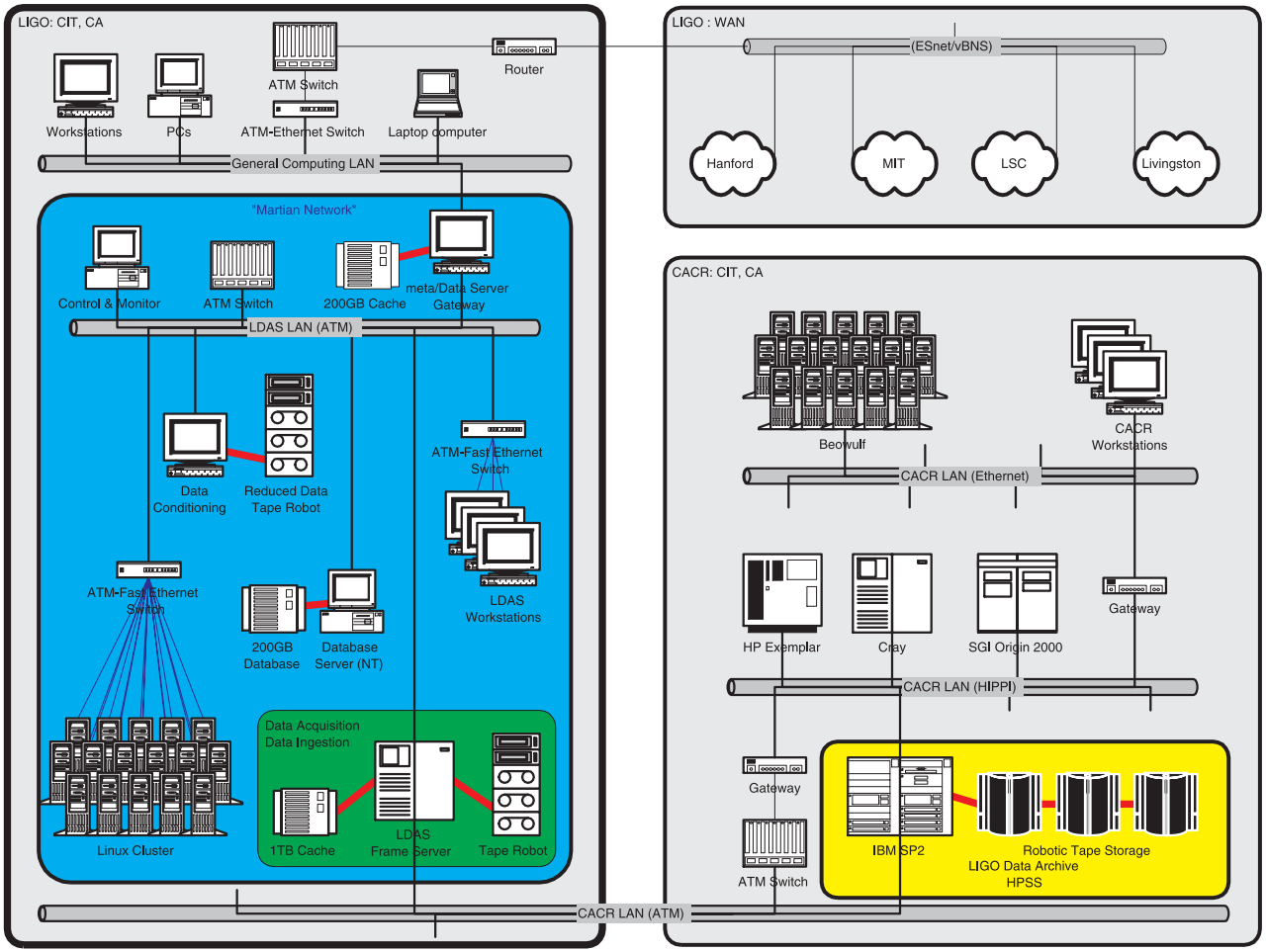
System Design Milestones

- System Specification
  - *LIGO Data Analysis White Paper*: LIGO-M970065-B-E
  - June, 1997
- Design Requirements Review
  - *LDAS Design Requirements*: LIGO-T970159-4-E
  - *LDAS Conceptual Design*: LIGO-T970160-6-E
  - December, 1999
- Preliminary Design Review
  - *LDAS Preliminary Design*: LIGO-T990001-6-E
  - March, 1999

Hardware Architecture(site)

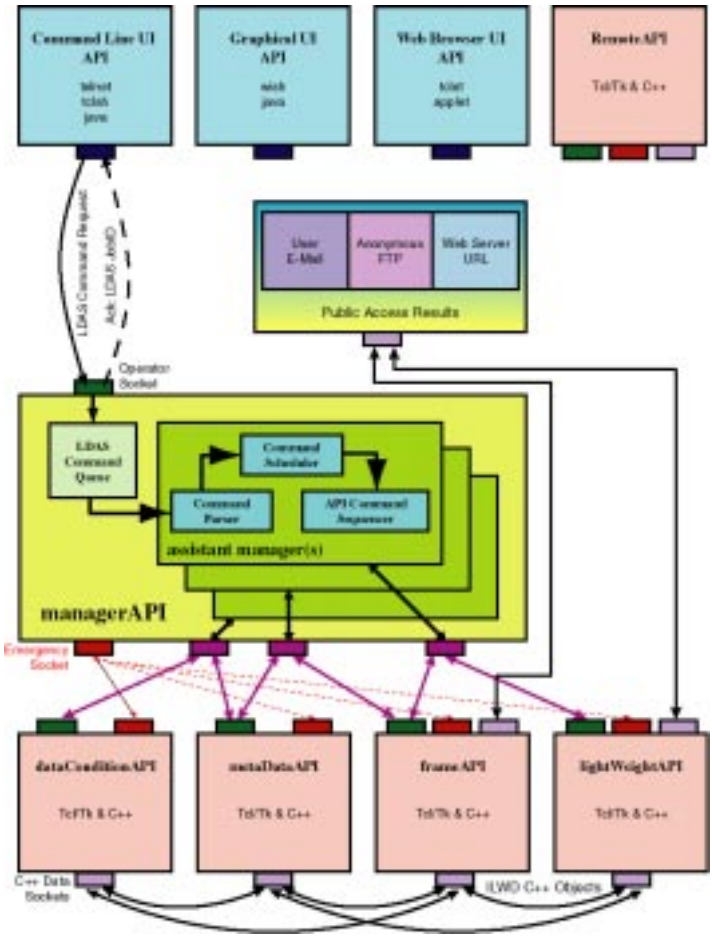


Hardware Architecture (caltech)



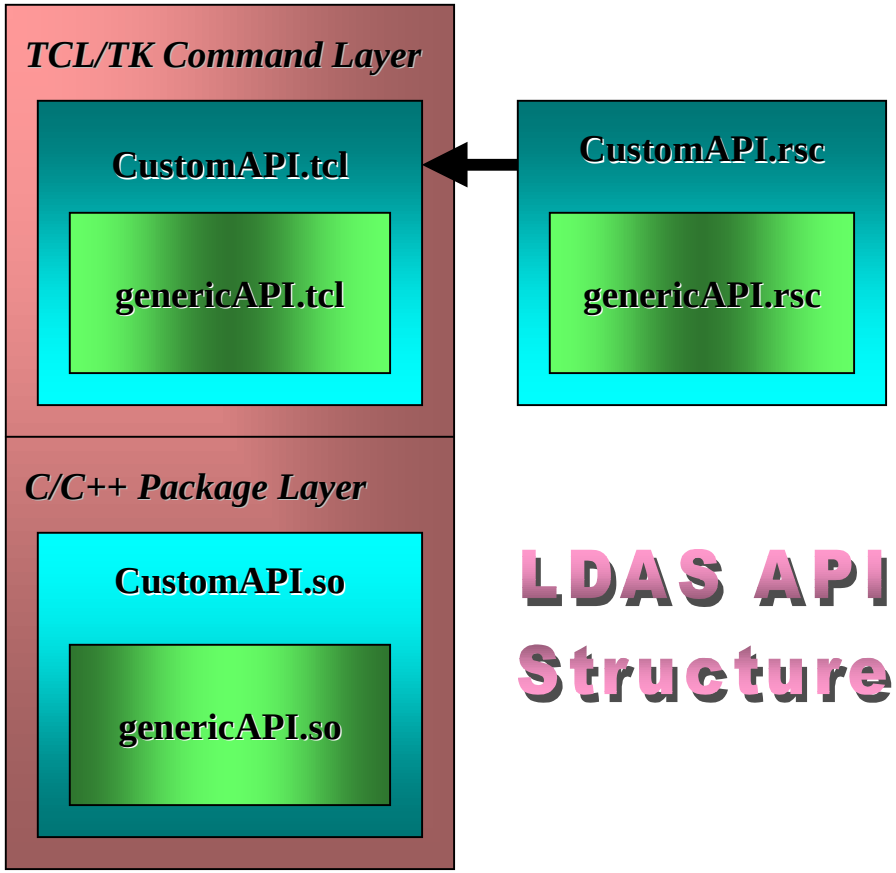
Client - Server - Servlets

- Clients: User Interfaces
  - Three types of UIs:
    - command line
    - graphical
    - web browser
  - Send LDAS Command requests to “Operator Socket” on managerAPI and receive acknowledgement containing “jobID” and other details associated with request.
- Server: managerAPI
  - Uses 3-10 assistant managers to handle each LDAS Command Request independently.
- Servlets: LDAS APIs
  - Internal LDAS APIs & RemoteAPI using LDAS interfaces.
  - Analysis flow control managed by “meta-scripts” associated with LDAS Commands.
  - Data communicated internally as ILWD objects
  - Data shared externally as Frames & LIGO_LW(XML).



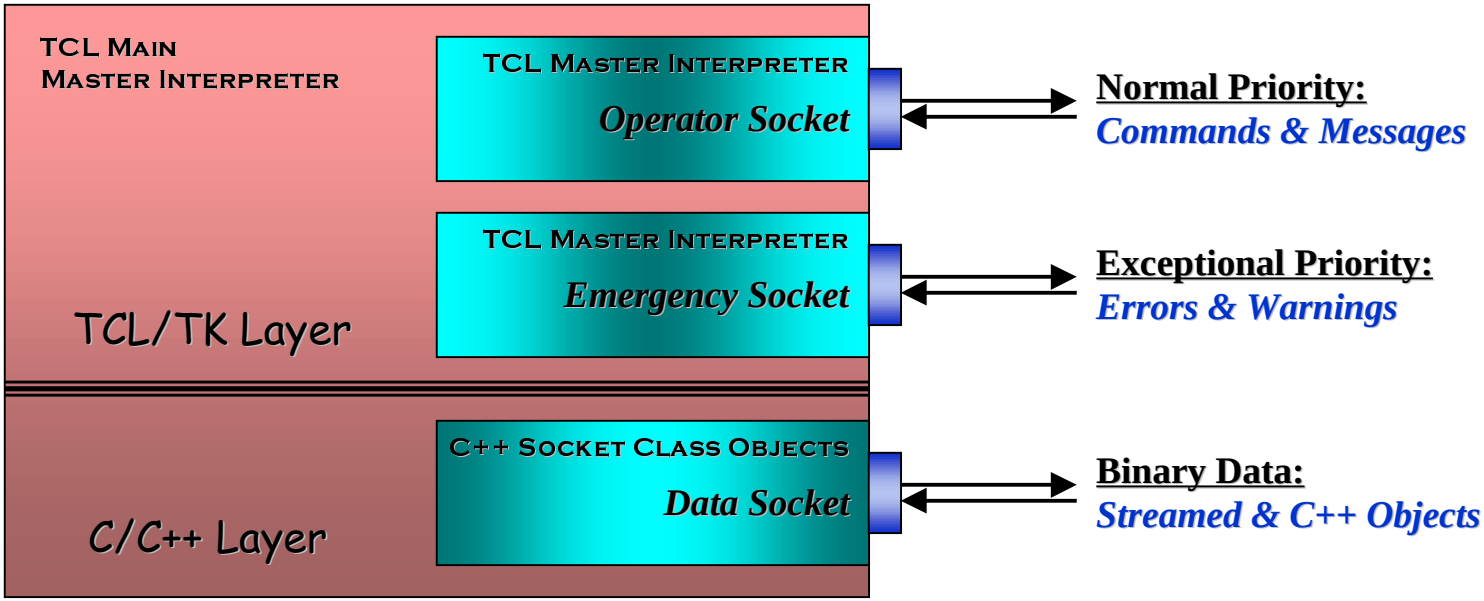
Layered APIs

- LDAS APIs:
  - Two Primary Layers:
    - Tcl/Tk (glue code)
    - C/C++ (Tcl extensions)
    - Use Swig to interface
  - GenericAPI (core) Module:
    - Communications
      - TCL - C++ using SWIG
      - API - API using sockets
    - Common Tcl proc functions:
      - Logging
      - Operator/emergency socket management
      - Resource/configuration management
    - Common C/C++ extensions:
      - Data socket management
      - ILWD data objects
      - Object serialization
      - object save/restore methods
  - CunstomAPI (specialization) Modeule:
    - unique/modular tasks:
      - database, frames , ligo_lw, data conditioning, parallel computing, etc.



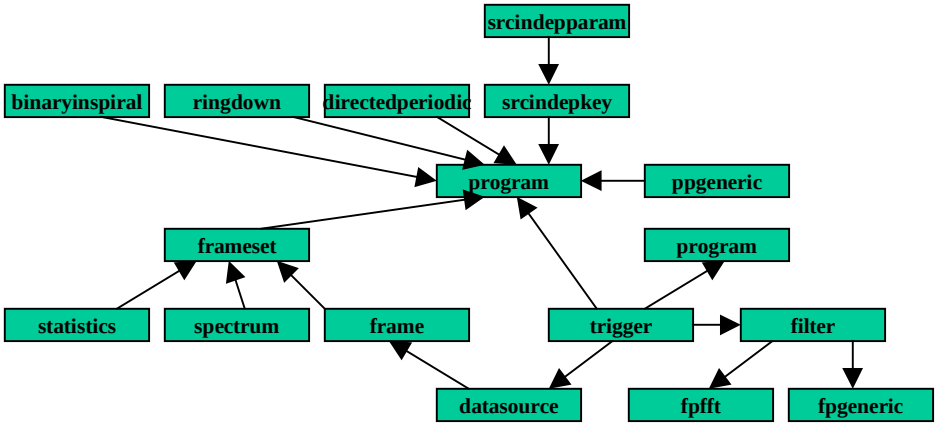
API Socket Communications

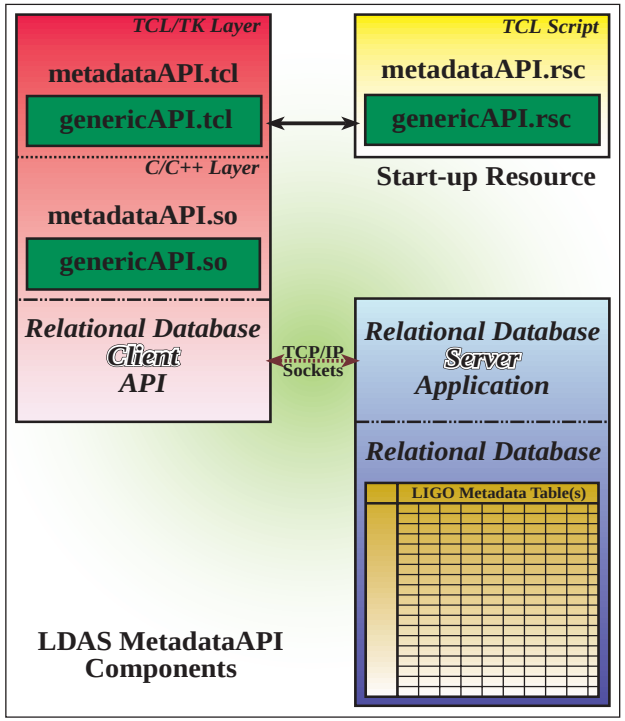
- 3 Types of socket communications in each LDAS API:
  - Operator Sockets - Normal inter-process *commands* & *messages* (Tcl Layer)
  - Emergency Sockets - Error & warning *commands* & *messages* (Tcl Layer)
  - Data Sockets - Binary Data in *raw streams* or *C++ objects* (C++ Layer)



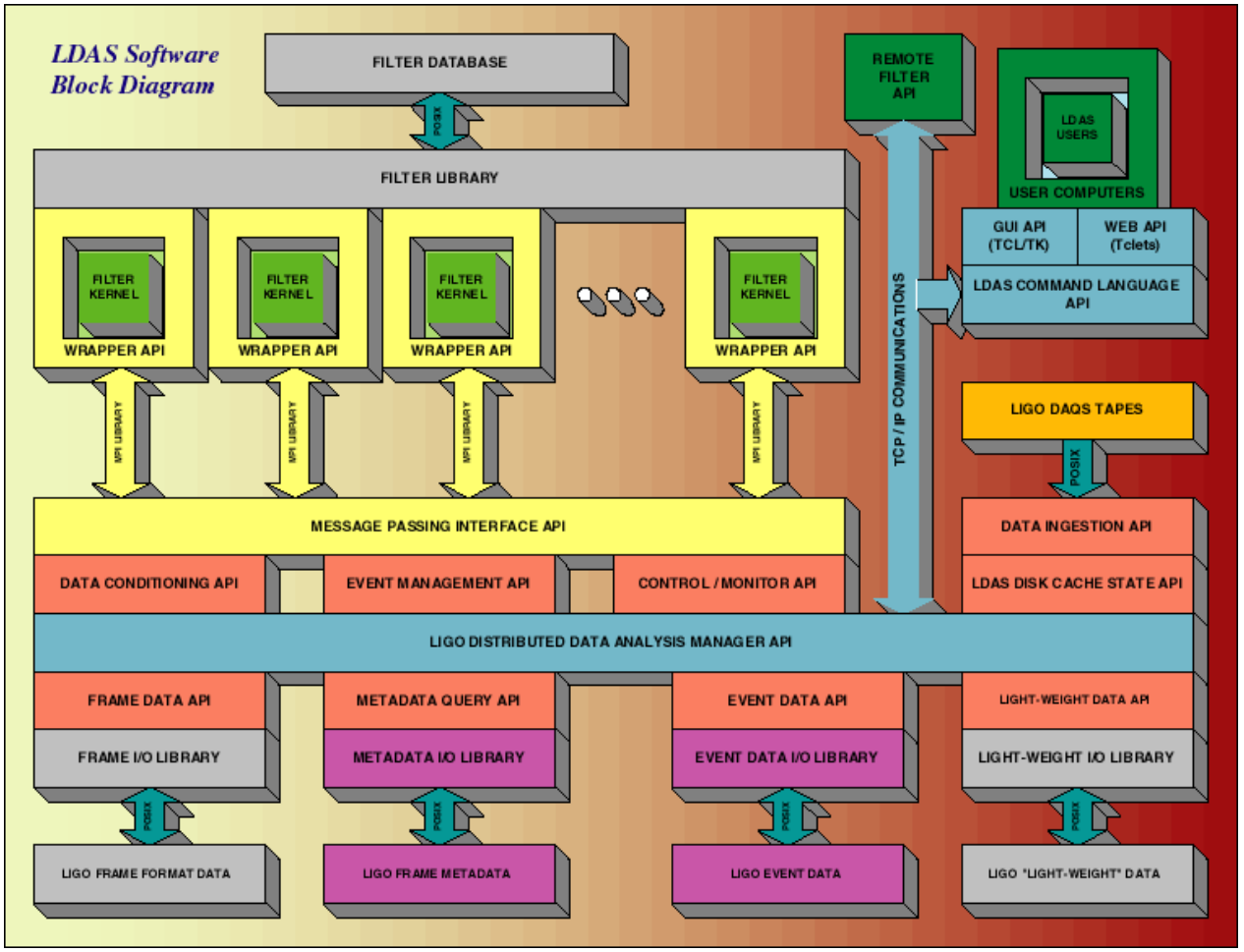
Distributed Database API

- metaDataAPI:
  - Relational Database
    - Using ODBC standard for connectivity
    - Chose IBM's DB2
      - Currently using 5.2, plan upgrade to 6.1
    - DB2 server can be on Solaris or Linux now
    - Table contents
      - Centralized around program table
      - Frame tables
      - Astrophysical event tables
      - GDS diagnostic trigger tables
    - Table relationships





Software Block Diagram



Site Installation

- Hanford
  - June 20th - June 25th, 1999:
    - *Initial Hardware Installation*
  - July 25th - July 30th, 1999:
    - *Initial Software Installation*
  - September 26th - November 1st, 1999:
    - *Continued Software & Hardware Installation*
- Livingston
  - October 24th - October 29th, 1999:
    - *Initial Hardware Installation*

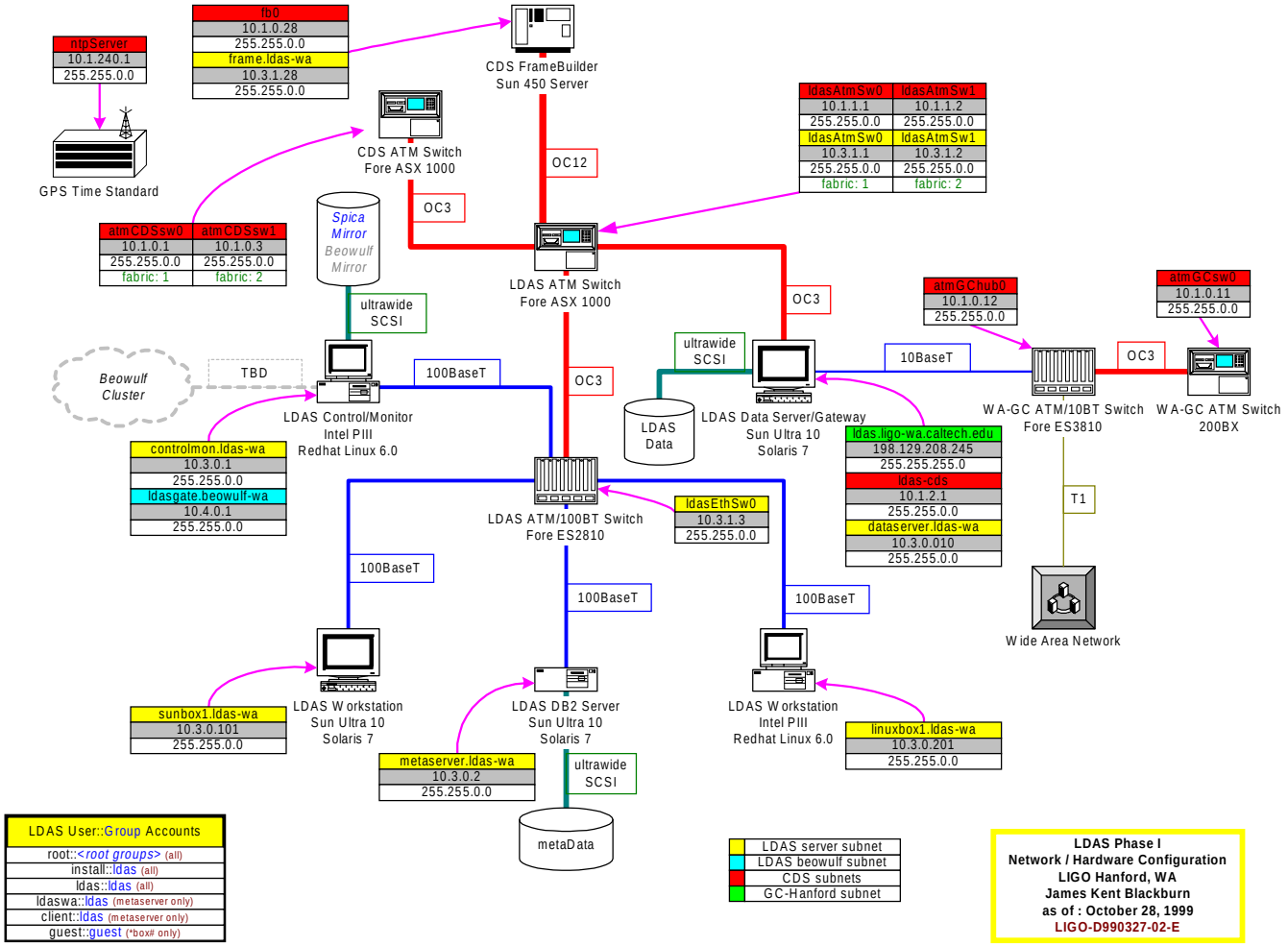
Hanford Hardware Rack

- Hanford: Jun, Jul, Sep, 99
  - **dataserver:** ***managerAPI / frameAPI***
    - **Sun Ultra 10 + 18GB HD**
  - **metaserver:** ***metadataAPI***
    - **Sun Ultra 10 + 18GB HD**
  - **controlmon:** ***internal software mirror***
    - **Pentium III PC + 18GB HD**
  - **Fore 1000 ATM switch**
  - **Fore 2810 fast ethernet switch**
  - **1 sun & 1 linux workstations**
    - **Sun Ultra10**
    - **Pentium III PC**
  - **video/keyboard switch**

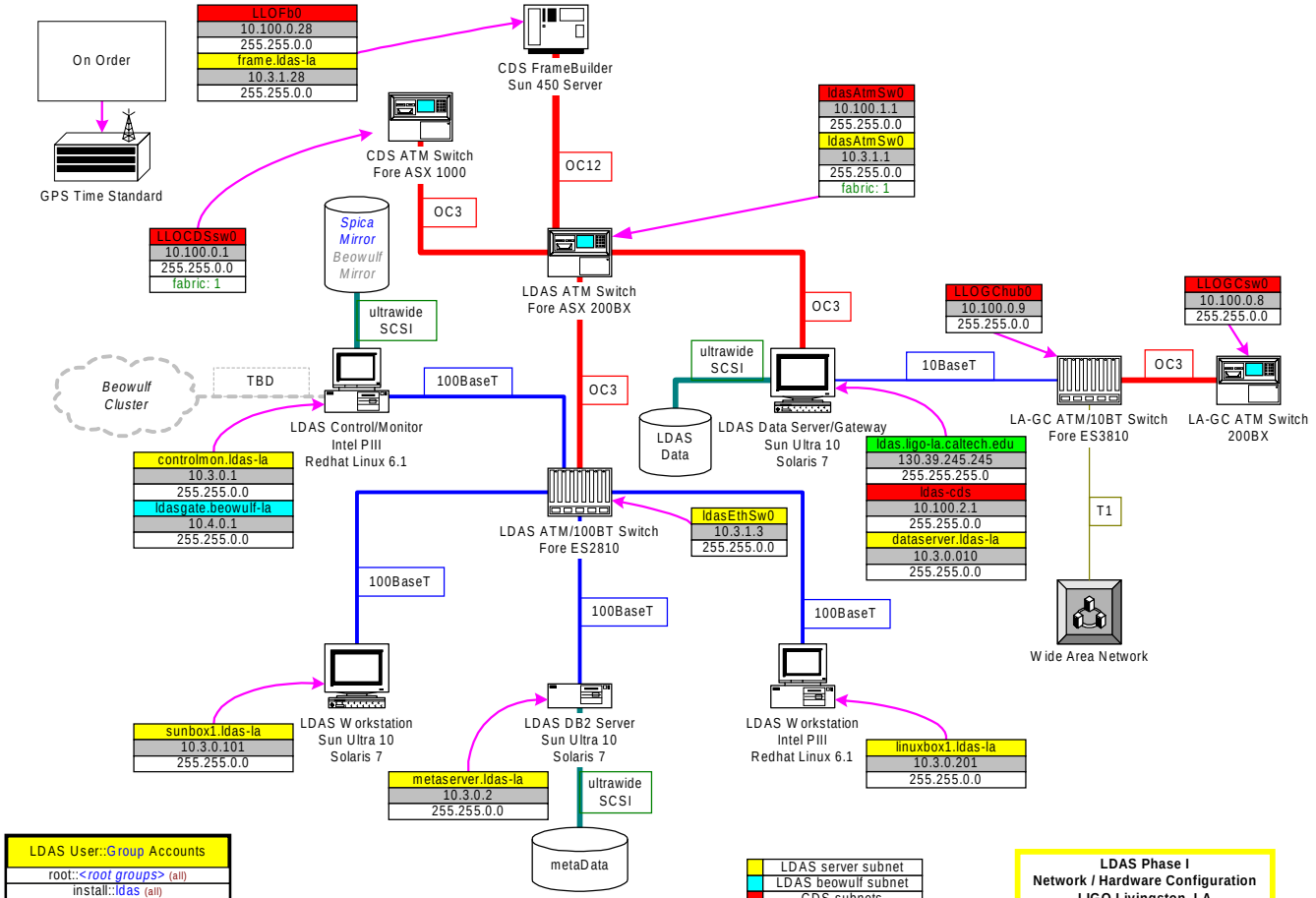
Livingston Rack

- Livingston: Oct, 99
  - **dataserver:** ***managerAPI / frameAPI***
    - **Sun Ultra10 + 18GB HD**
  - **metaserver:** ***metadataAPI***
    - **Sun Ultra10 + 18GB HD**
  - **controlmon:** ***internal software mirror***
    - **Pentium III PC + 18 GB HD**
  - **Fore 200BX ATM switch**
  - **Fore 2810 fast ethernet switch**
  - **1 sun & 1 linux workstations**
    - **Sun Ultra10**
    - **Pentium III PC**
  - **video/keyboard switch**

Hanford Network Diagram



Livingston Network Diagram



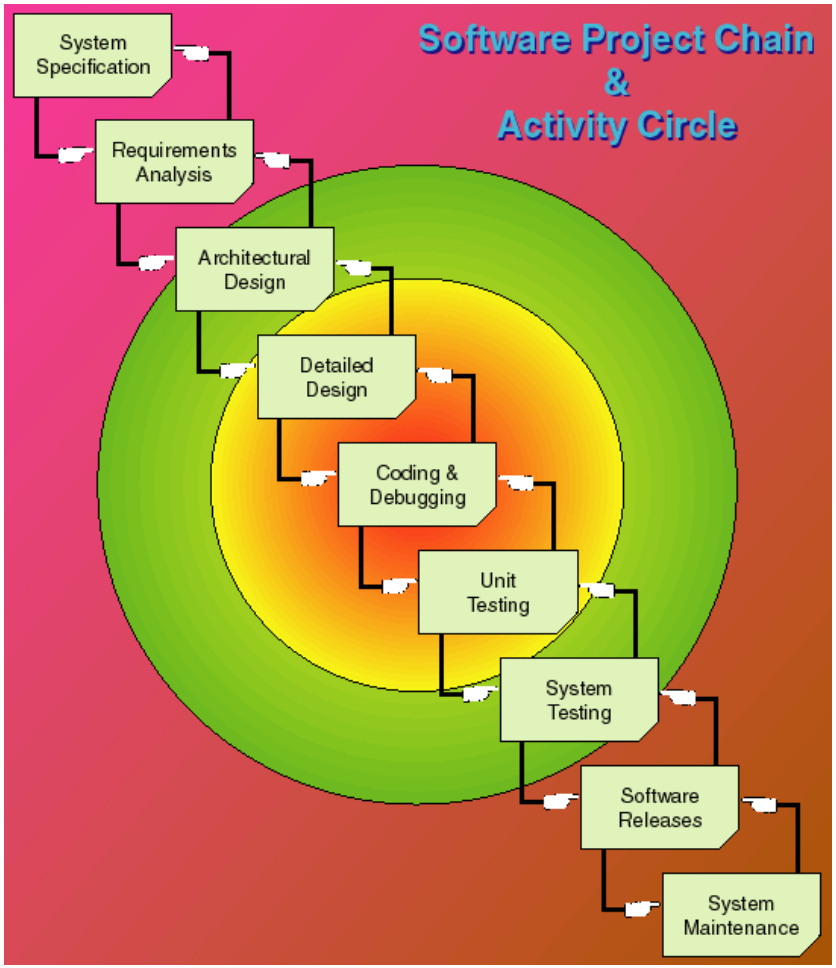
Site Software Installation

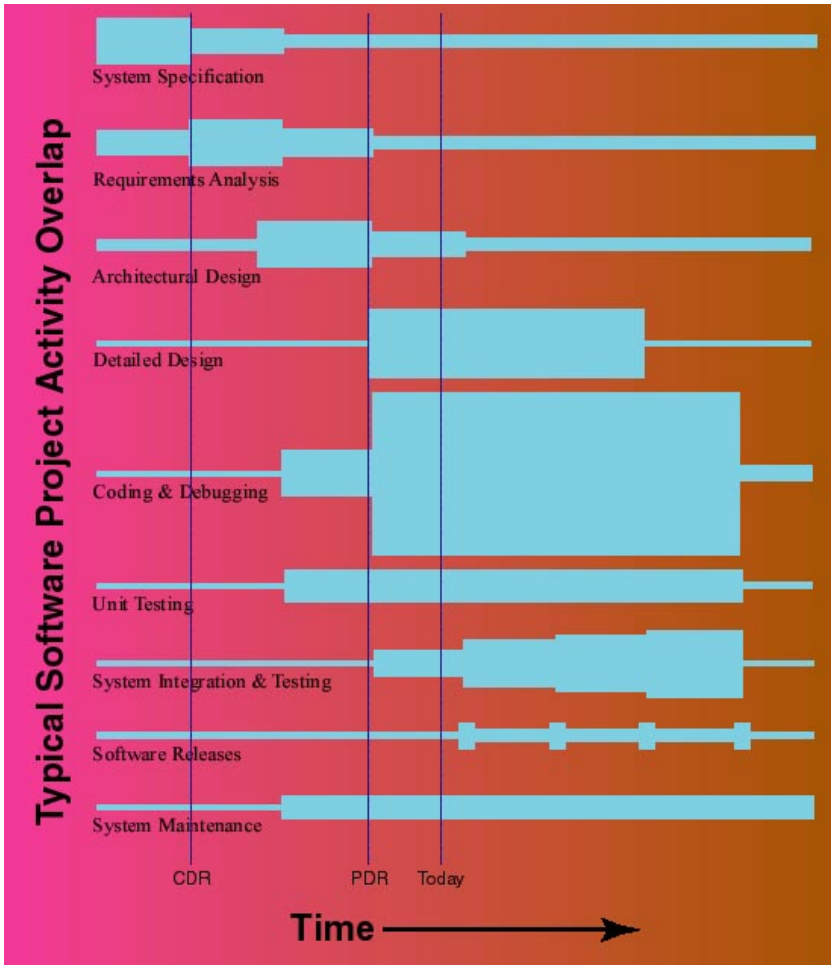
- Development Server: spica.ligo.caltech.edu
  - compilers, interpreters, debuggers, editors, etc
  - LDAS CVS Repository
  - LDAS release installations
  - all software managed using "*stow*" utility
  - Soon: web based bug/change report system (*bugzilla* being considered)
- Spica mirrored to sites using "rsync" utility.
- Unix configurations managed with "cfengine" utility.
- Limited unix accounts: {install, ldas, ldaswa, client, guest, reboot, & root}
- Internet gateway provides access to LDAS "private net."
  - LDAS managerAPI, frameAPI, lightWeightAPI
  - anonymous ftp
  - web service
  - ssh services
- Account logins using "ssh" & "scp" utilities only!

Software Construction Status

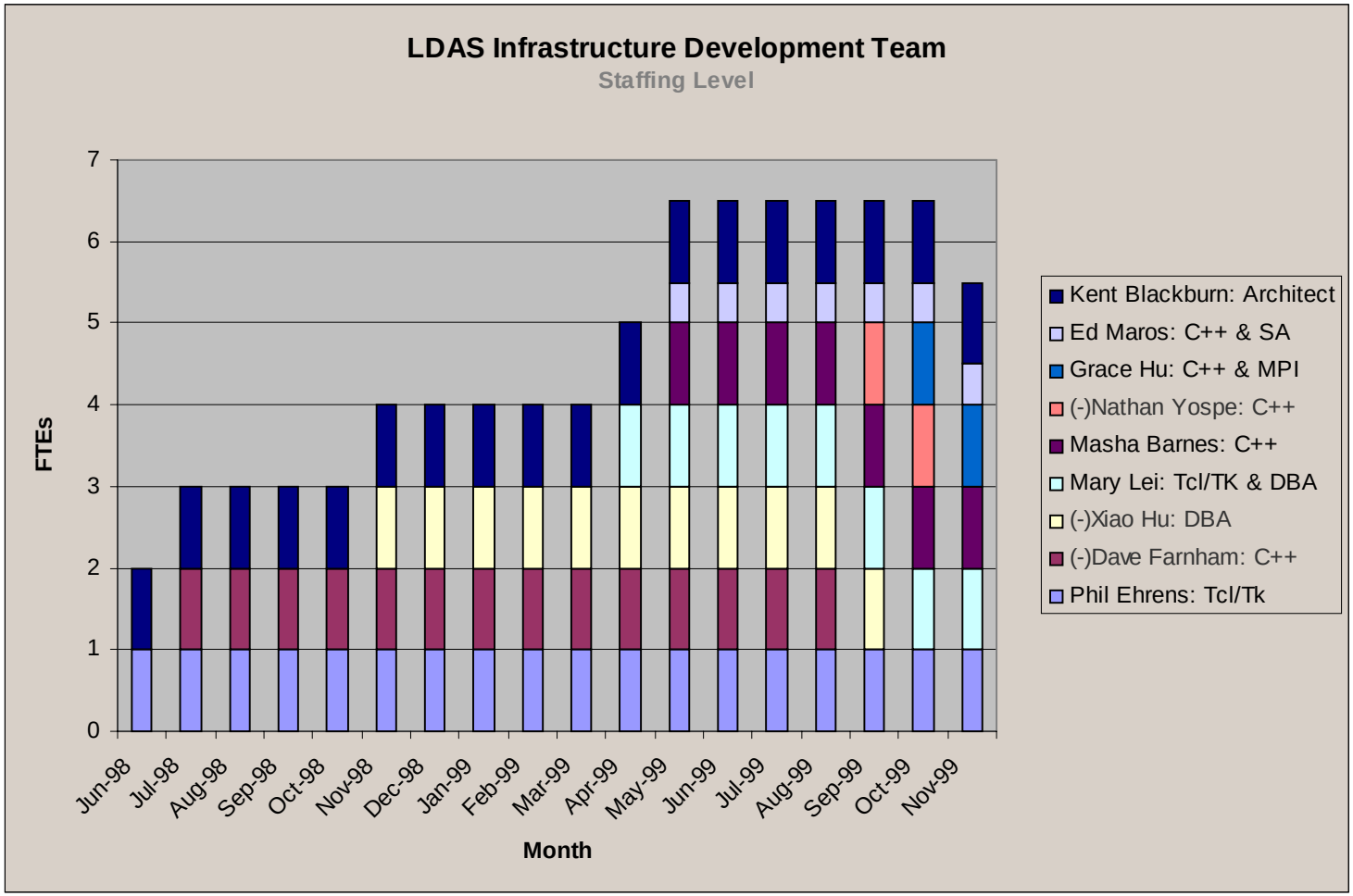
- Software Development Cycle
- Staffing
- Progress to Date
- CVS Repository Details
  - C++ Object Oriented Libraries
  - C++ API Shared Objects
  - Tcl/Tk Scripts

Software Project Activities

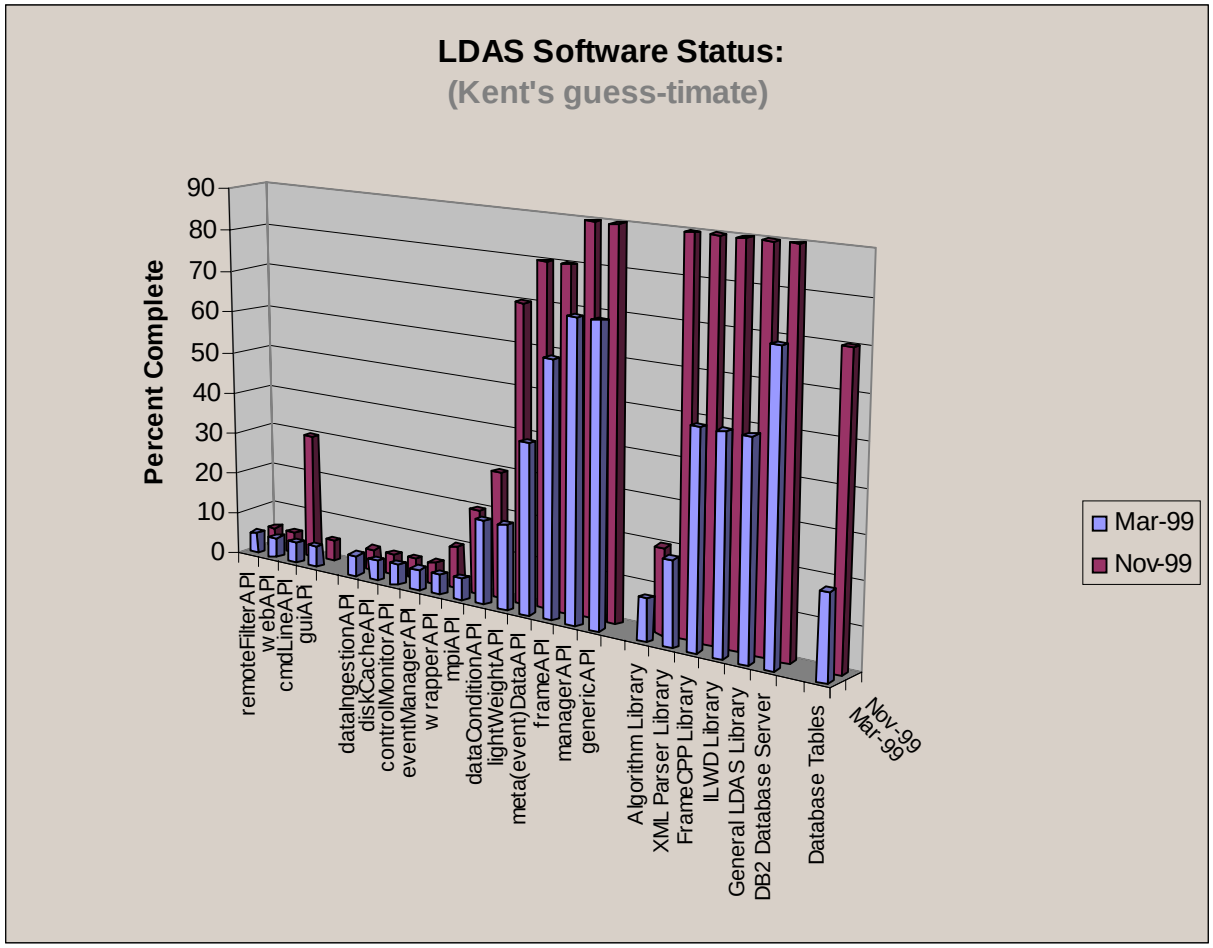




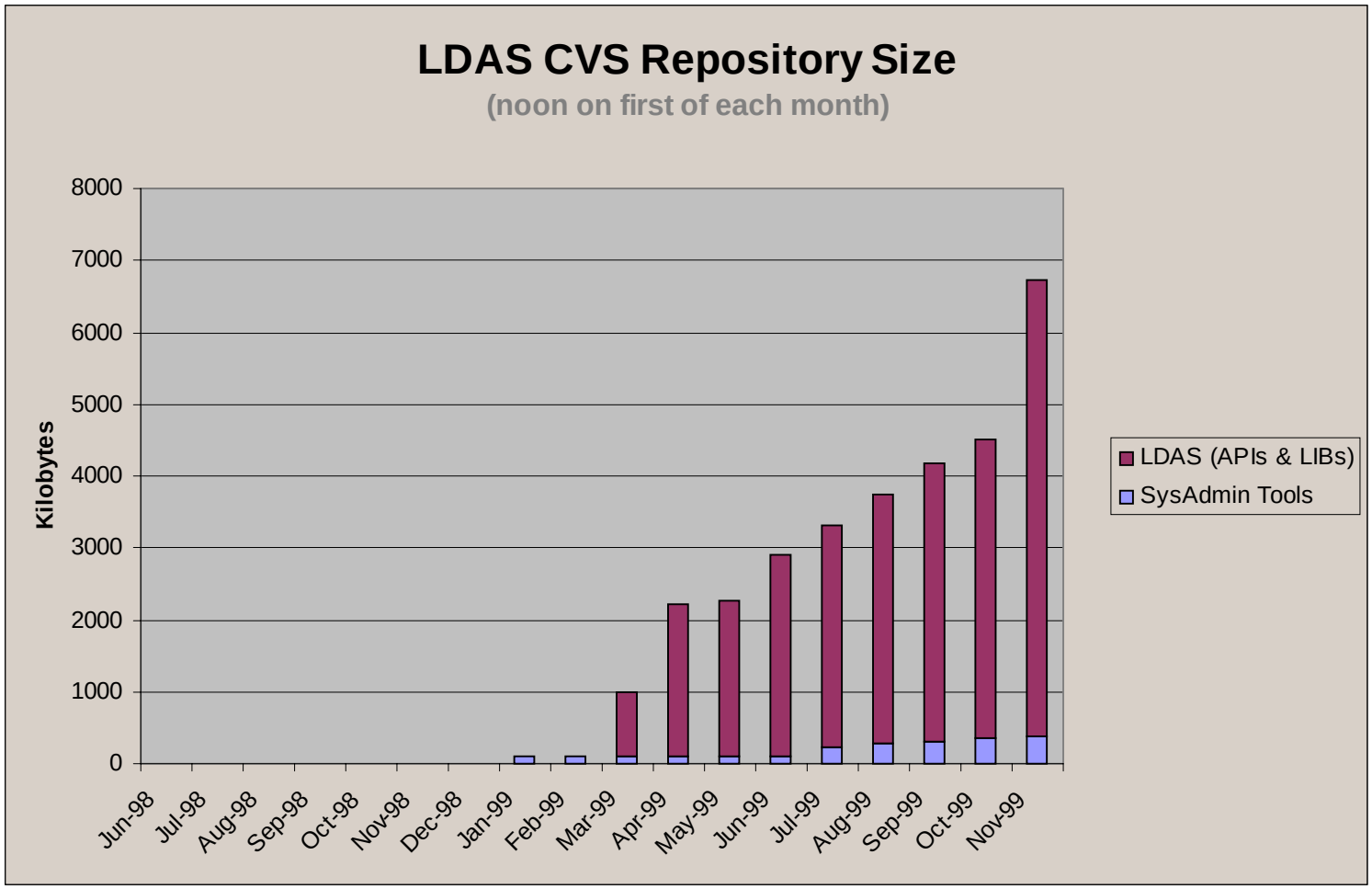
LDAS Software Team



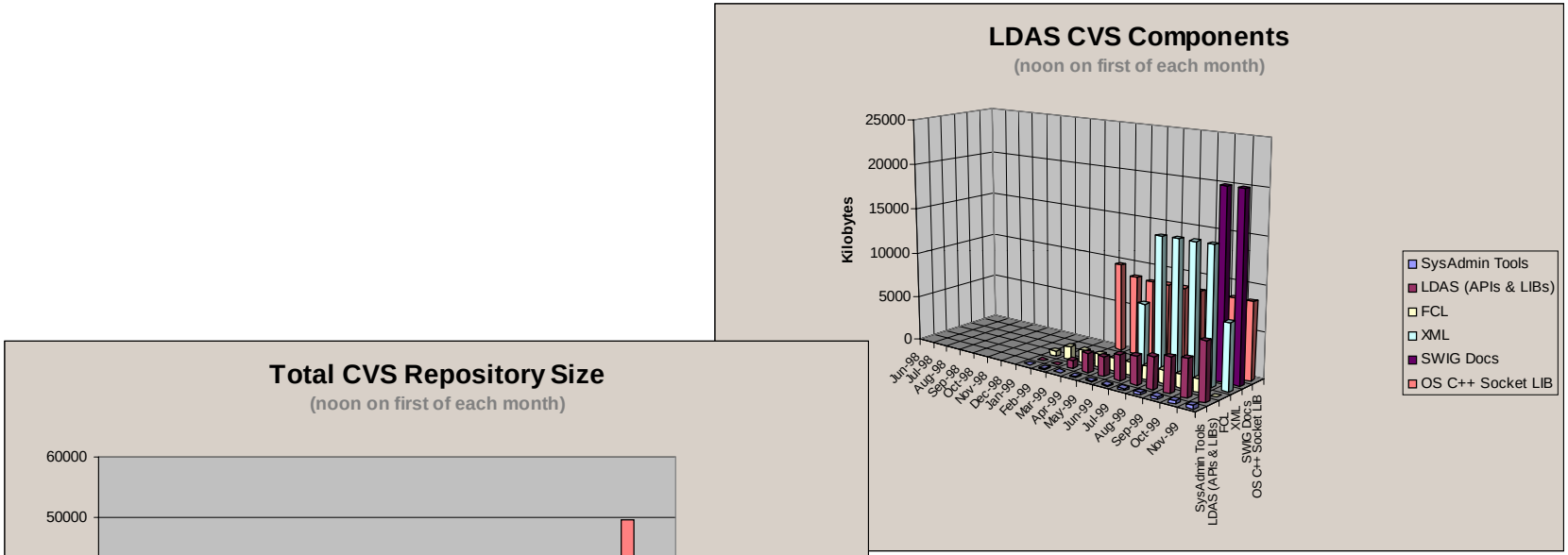
Software Status Estimate

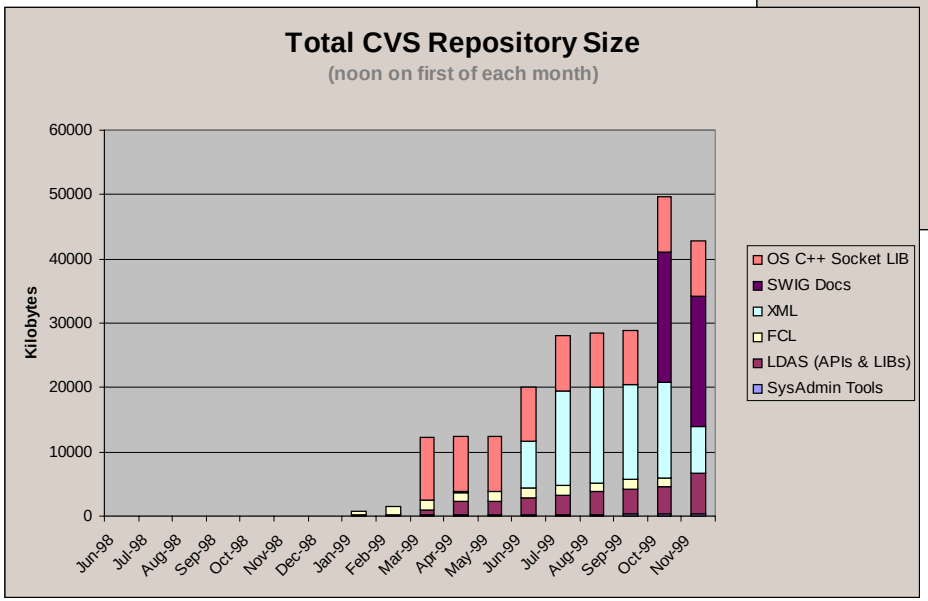


New LDAS Software

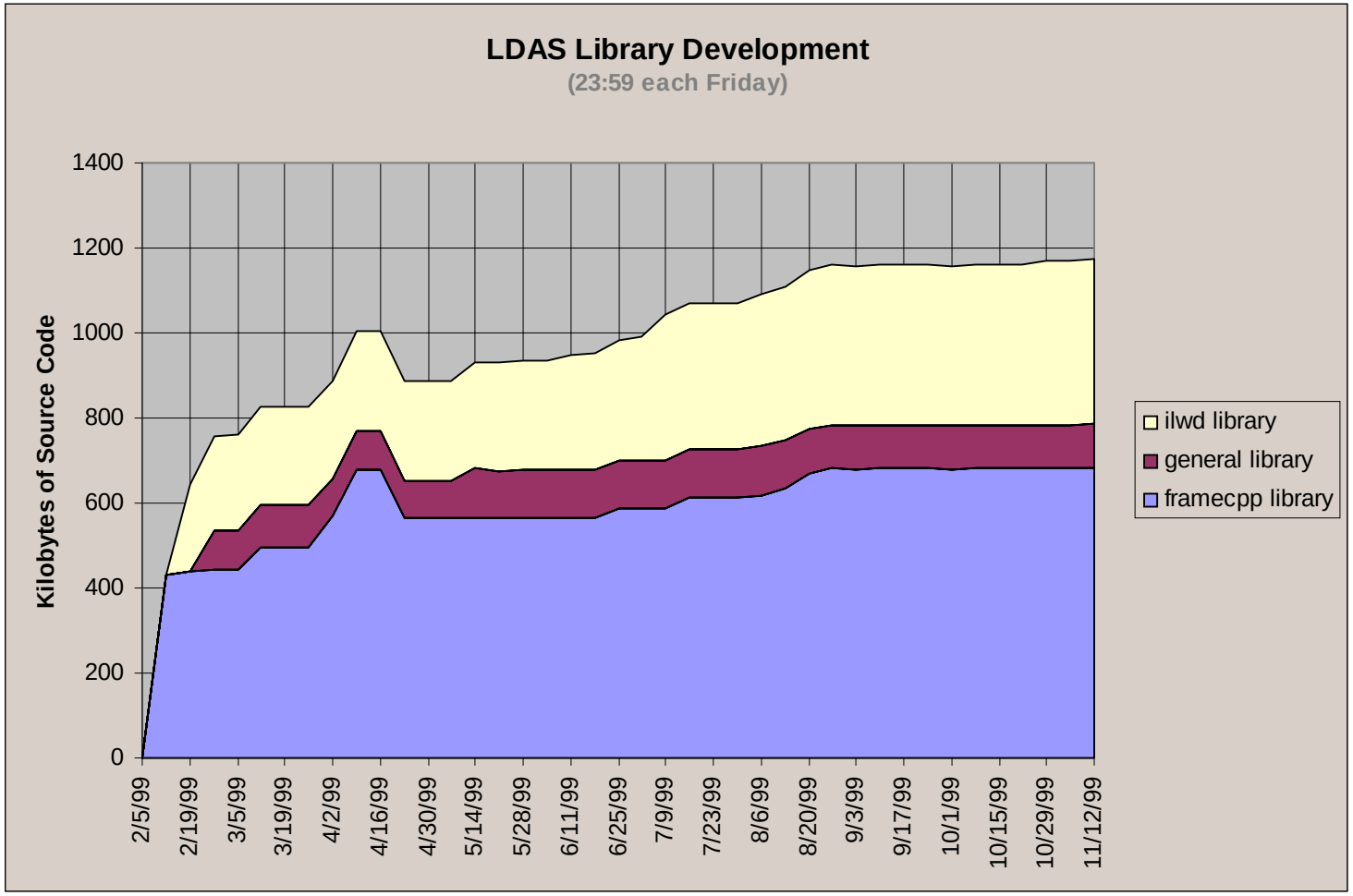


CVS Repository Details

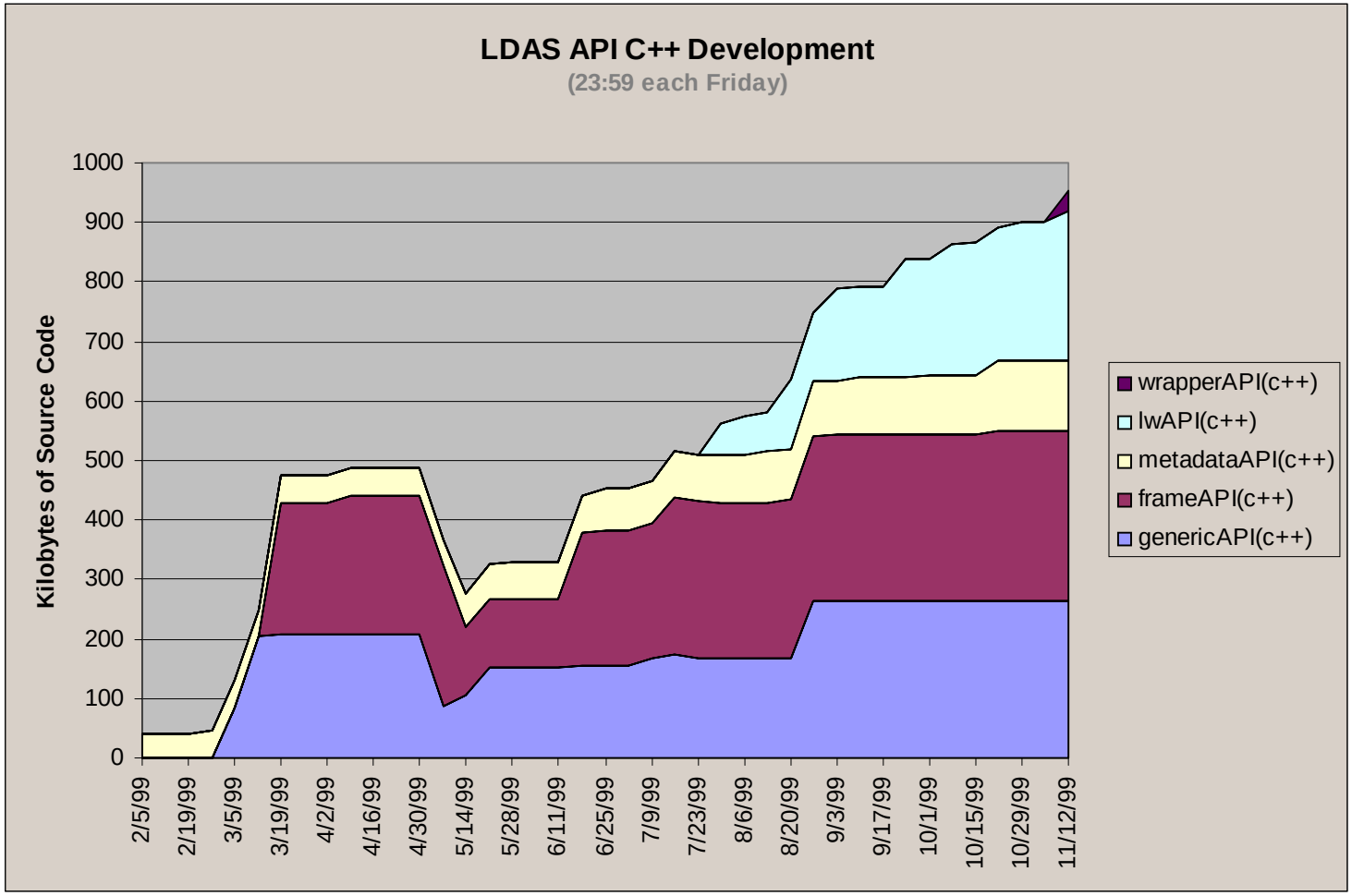




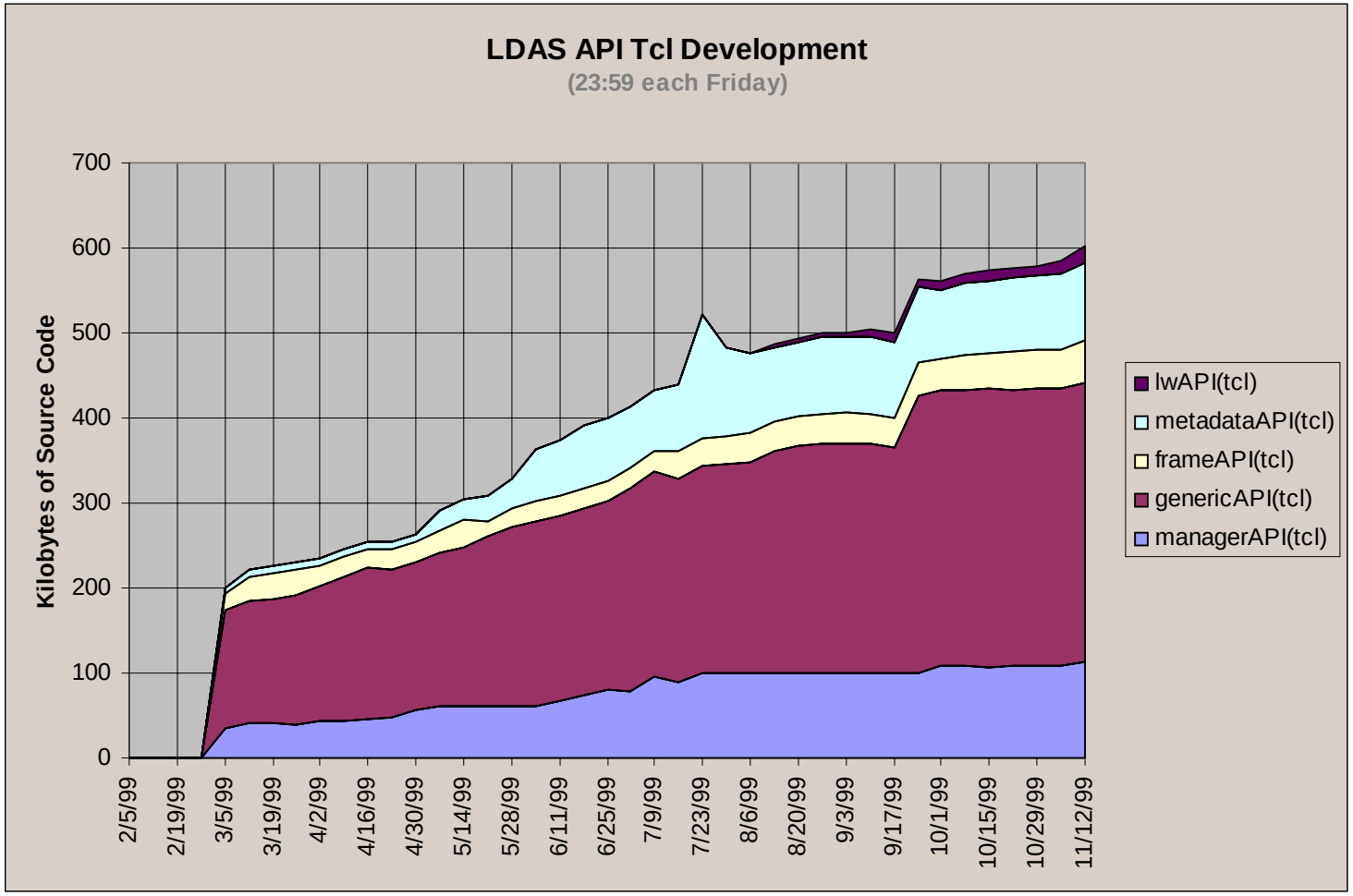
C++ Library Code



C++ API Shared Object Code



Tcl API Scripting Code

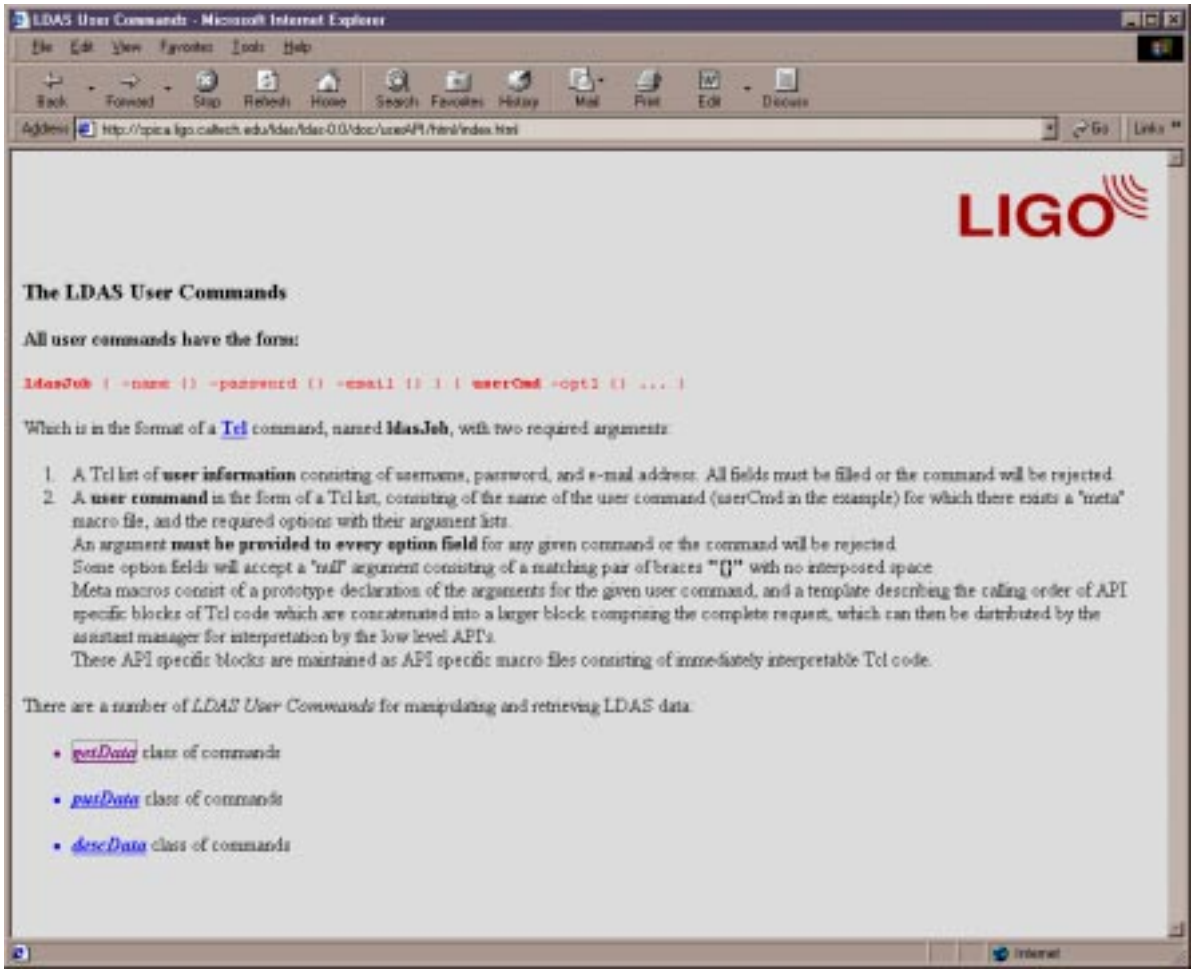


Software Documentation

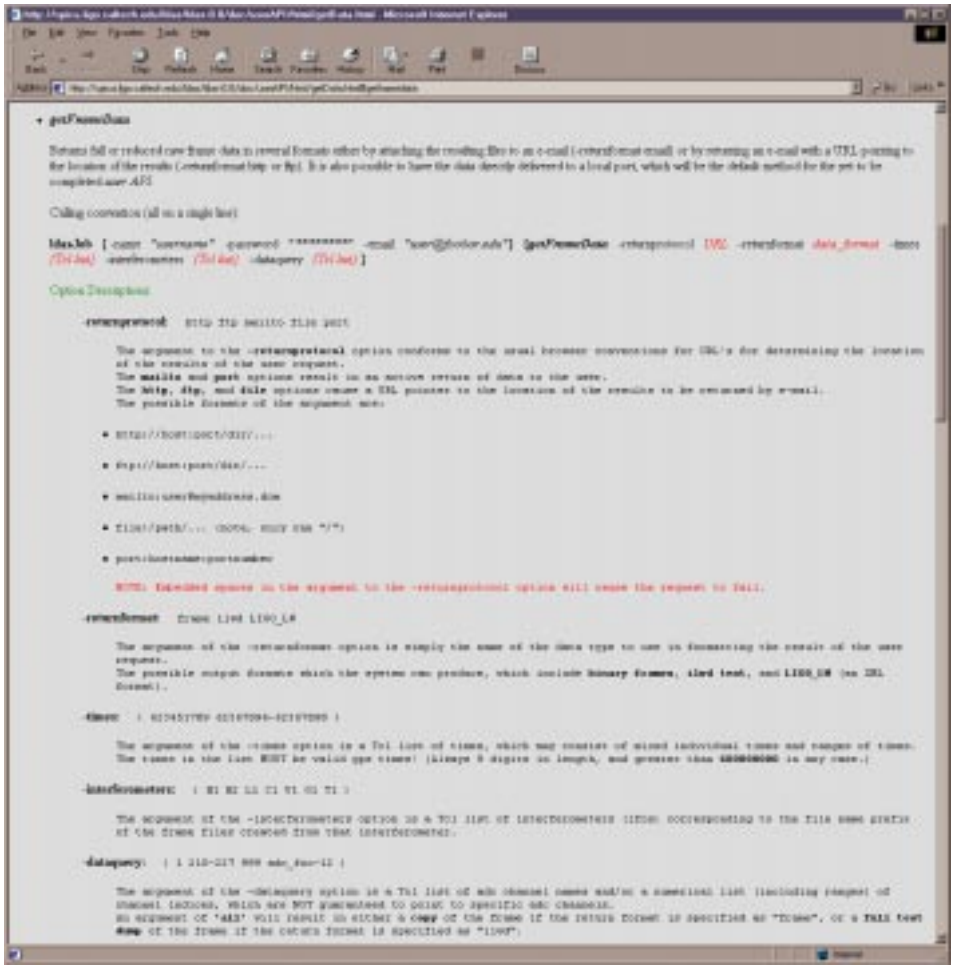
- Web Documentation
  - **Caltech:** http://www.ldas.ligo.caltech.edu/ldas/ldas-0.0/doc/index.html
  - **Hanford:** http://www.ldas.ligo-wa.caltech.edu/ldas/ldas-0.0/doc/index.html
  - **Livingston:** http://www.ldas.ligo-la.caltech.edu/ldas/ldas-0.0/doc/index.html
- User Interfaces
- Tcl/Tk - *TclDoc*
- C++ - *Perceps*
- Log Files - *LDAS APIs*

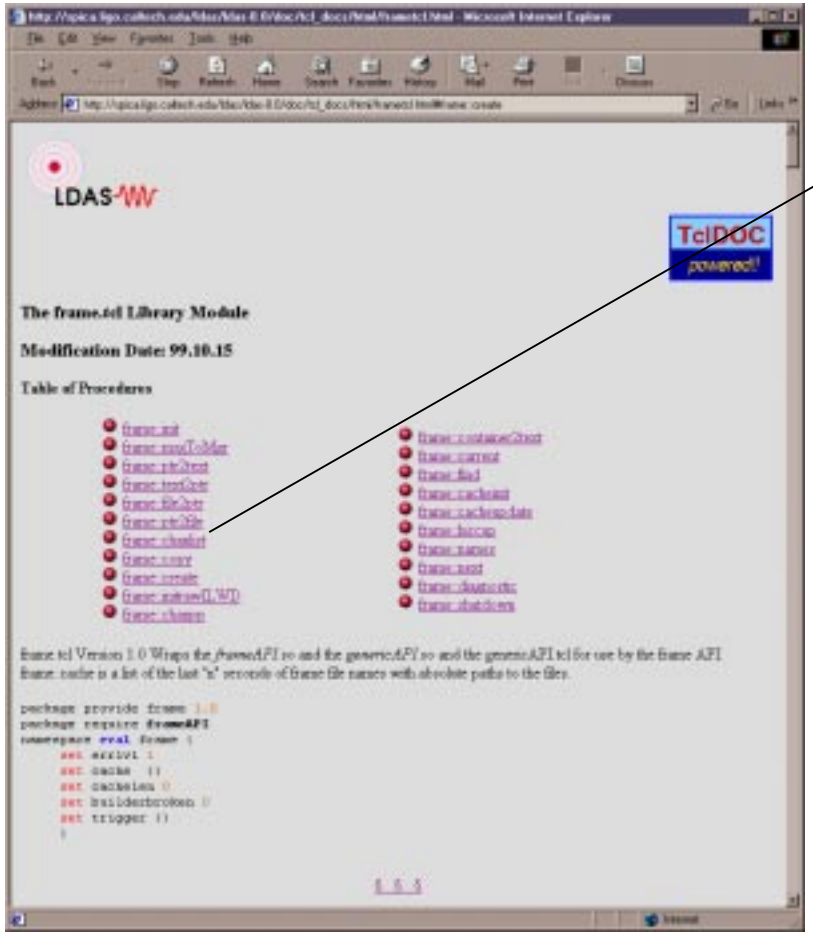
Software Home Page

User Command Model

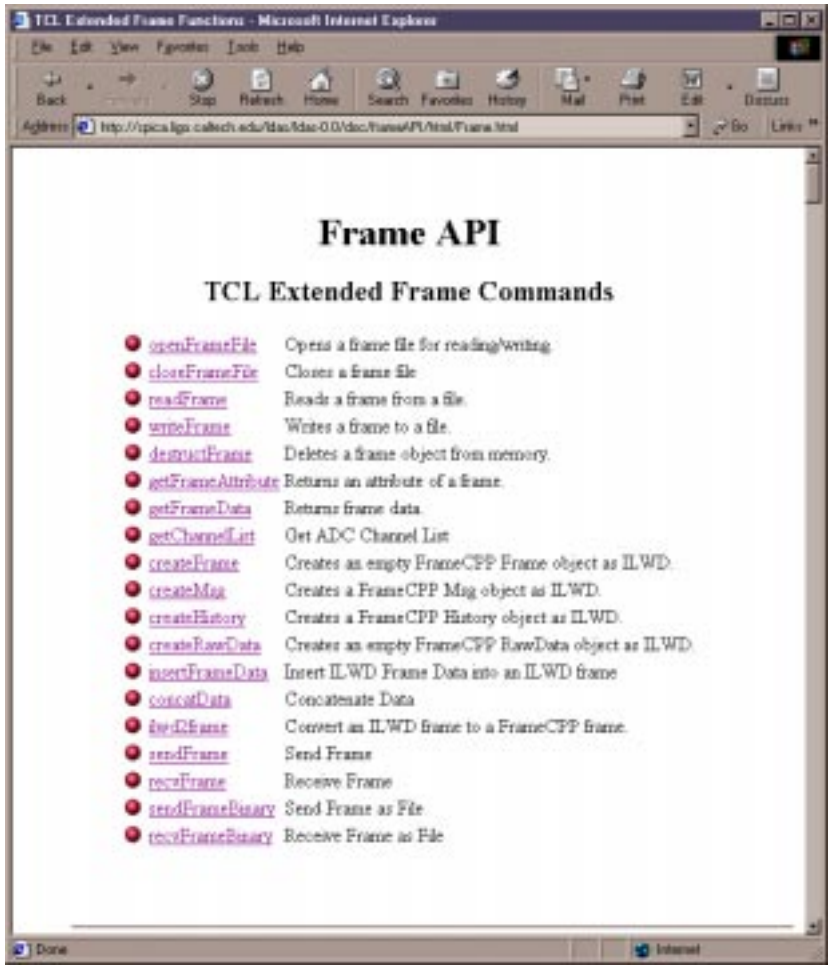


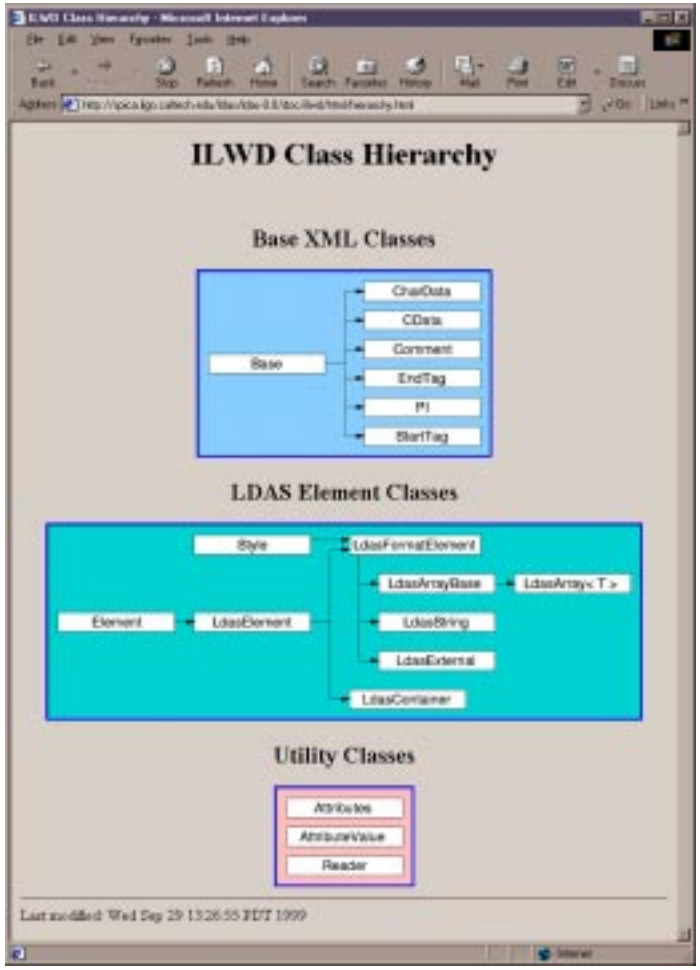
Examples of User Commands

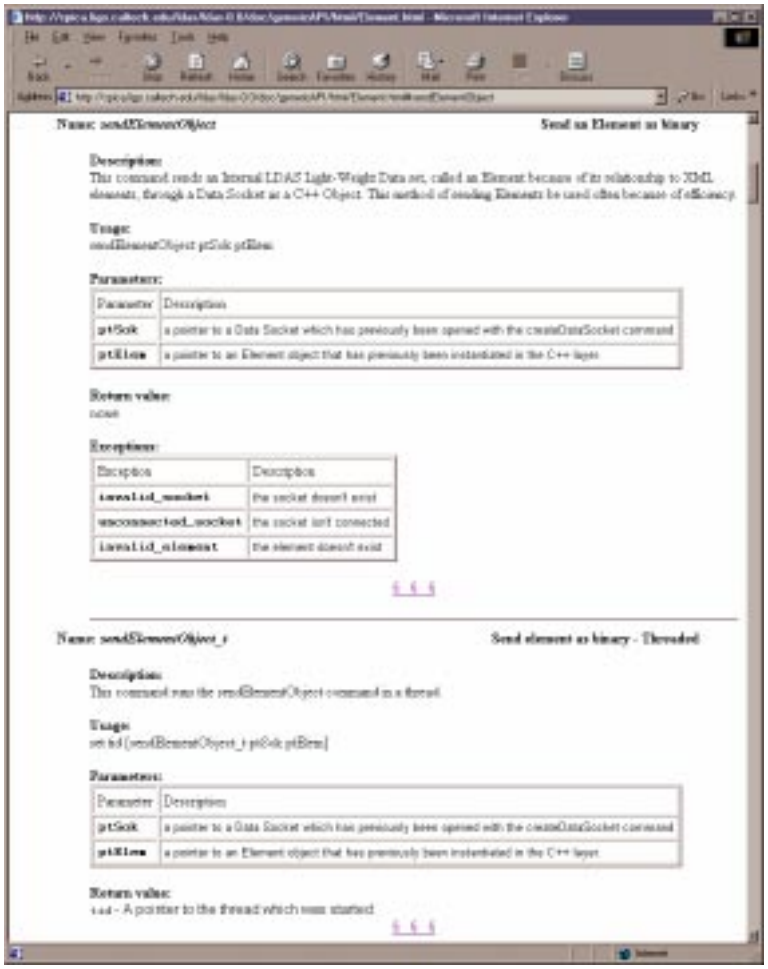



LDAS
TclDOC
powered!
The frame.tcl Library Module
Modification Date: 99.10.15
Table of Procedures

C++ Source Code Docs







Name: sendElementObject — Send an Element as binary

Description:
This command sends an Internal LDAS Light-Weight Data set, called an Element because of its relationship to XML elements, through a Data Socket as a C++ Object. This method of sending Elements be used often because of efficiency.

Usage:
sendElementObject ptSok ptElem

Parameters:

Parameter	Description
ptSok	a pointer to a Data Socket which has previously been opened with the createDataSocket command
ptElem	a pointer to an Element object that has previously been instantiated in the C++ layer

Return value:
none

Exceptions:

Exception	Description
invalid_socket	the socket doesn't exist
unconnected_socket	the socket isn't connected
invalid_element	the element doesn't exist

Name: sendElementObject_t — Send element as binary - Threaded

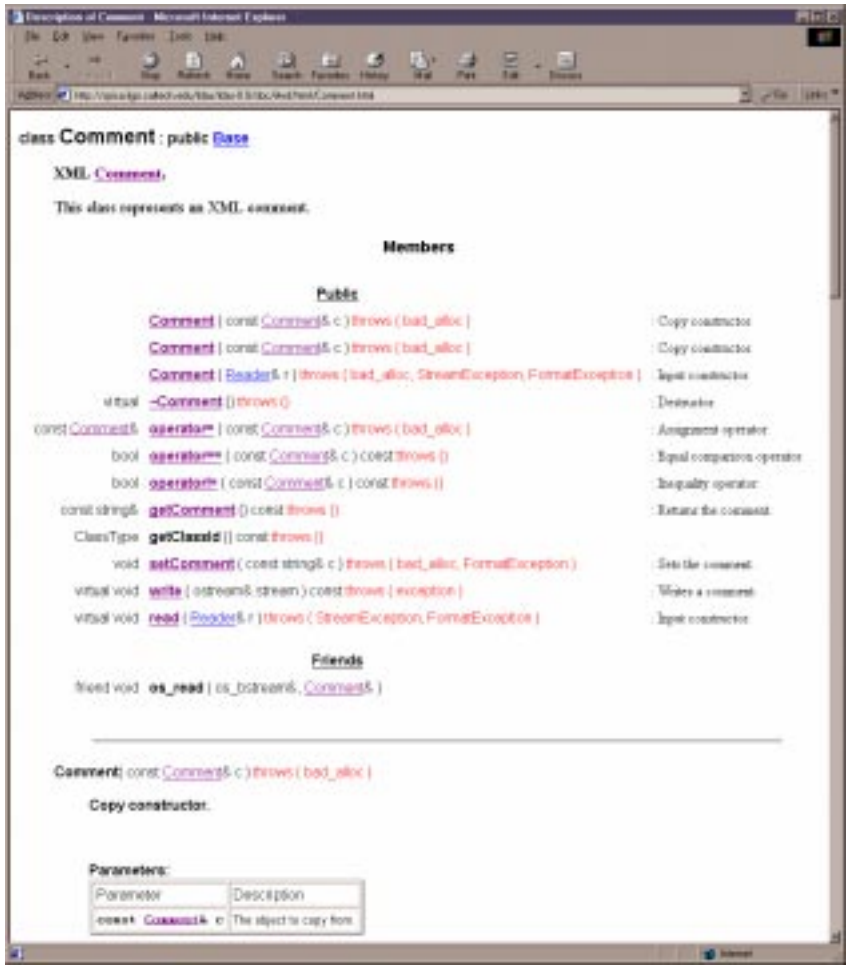
Description:
This command runs the sendElementObject command as a thread.

Usage:
set tid [sendElementObject_t ptSok ptElem]

Parameters:

Parameter	Description
ptSok	a pointer to a Data Socket which has previously been opened with the createDataSocket command
ptElem	a pointer to an Element object that has previously been instantiated in the C++ layer

Return value:
tid - A pointer to the thread which was started



class **Comment** : public Base

XML Comment.

This class represents an XML comment.

Members

Public

	Comment (const Comment& c) throw (bad_alloc)	Copy constructor
	Comment (const Comment& c) throw (bad_alloc)	Copy constructor
	Comment (Reader& r) throw (bad_alloc, StreamException, FormatException)	Input constructor
virtual	~Comment () throw ()	Destructor
const Comment&	operator= (const Comment& c) throw (bad_alloc)	Assignment operator
bool	operator== (const Comment& c) const throw ()	Equal comparison operator
bool	operator!= (const Comment& c) const throw ()	Inequality operator
const string&	getComment () const throw ()	Returns the comment
ClassType	getClassId () const throw ()	
void	setComment (const string& c) throw (bad_alloc, FormatException)	Sets the comment
virtual void	write (ostream& stream) const throw (exception)	Writes a comment
virtual void	read (Reader& r) throw (StreamException, FormatException)	Input constructor

Friends

friend void **os_read** (os_bstream&, Comment&)

Comment(const Comment& c) throw (bad_alloc)

Copy constructor.

Parameters:

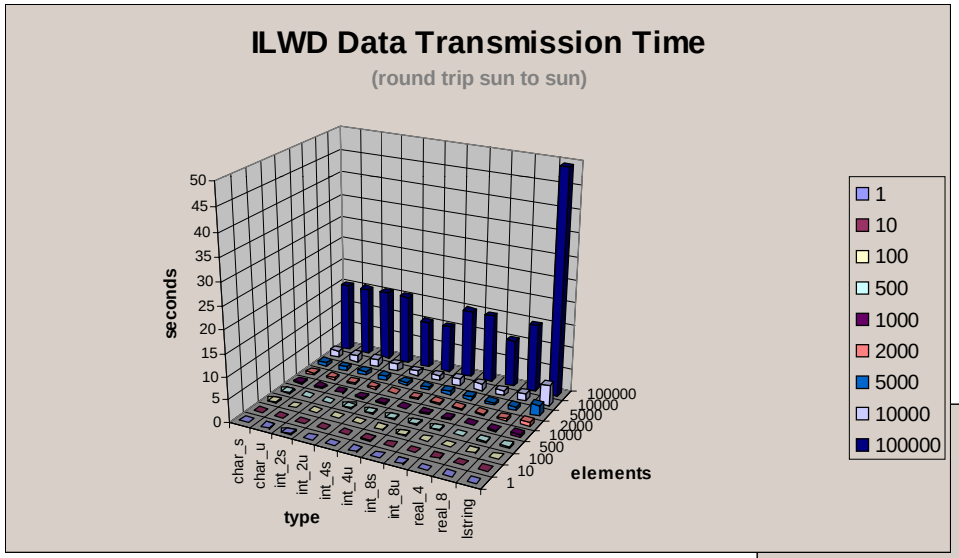
Parameter	Description
const Comment& c	The object to copy from

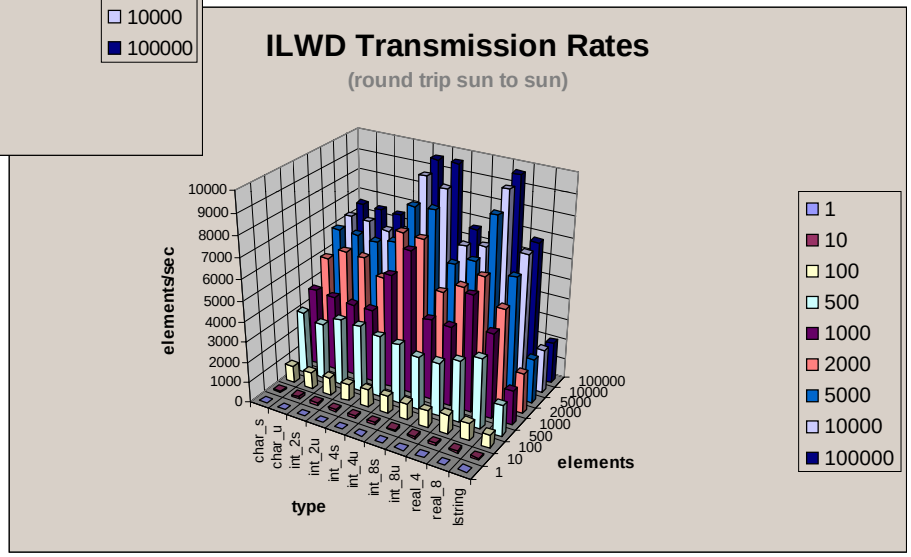
API Log Files

System Testing

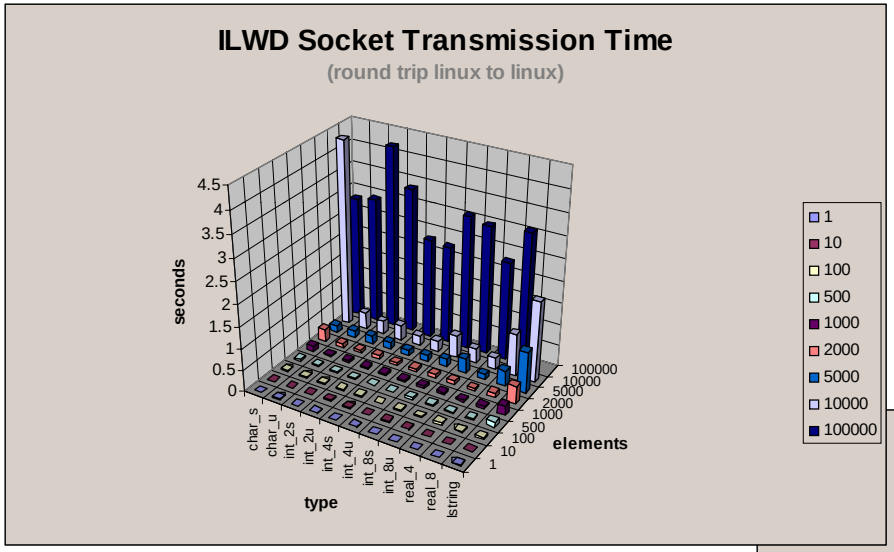
- ILWD Class Object Socket Communication
  - LDAS Internal Light Weight Data C++ Object Transmission Test
  - Uses two LDAS genericAPIs
  - Time to serialize C++ objects, send out socket, receive on other end, instantiate ILWD object, re-serialize object, send back to source, and re-instantiate ILWD object between two LDAS APIs
- Database Record Insertions Rates
  - LIGO Metadata Insertion Rates into DB2 tables
  - Uses managerAPI, metaDataAPI, OpenLink ODBC level 2.5 driver, and DB2 server
  - Times SQL inserts on “N” rows of each LIGO Table
- Frame to ILWD Translation Rates
  - LIGO Frame Files to ILWD translation rates for various channel cuts
  - Uses frameAPI and frames from Hanford’s framebuilder

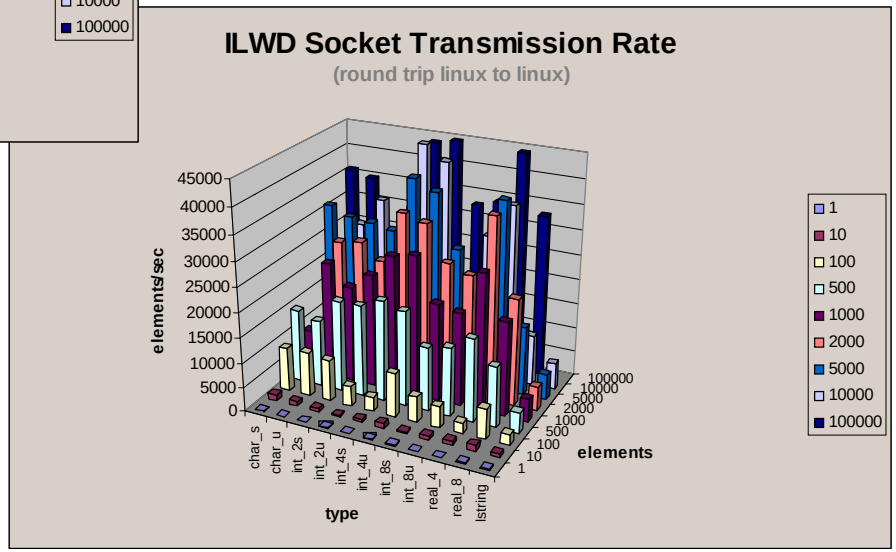
Sun Data Socket Tests



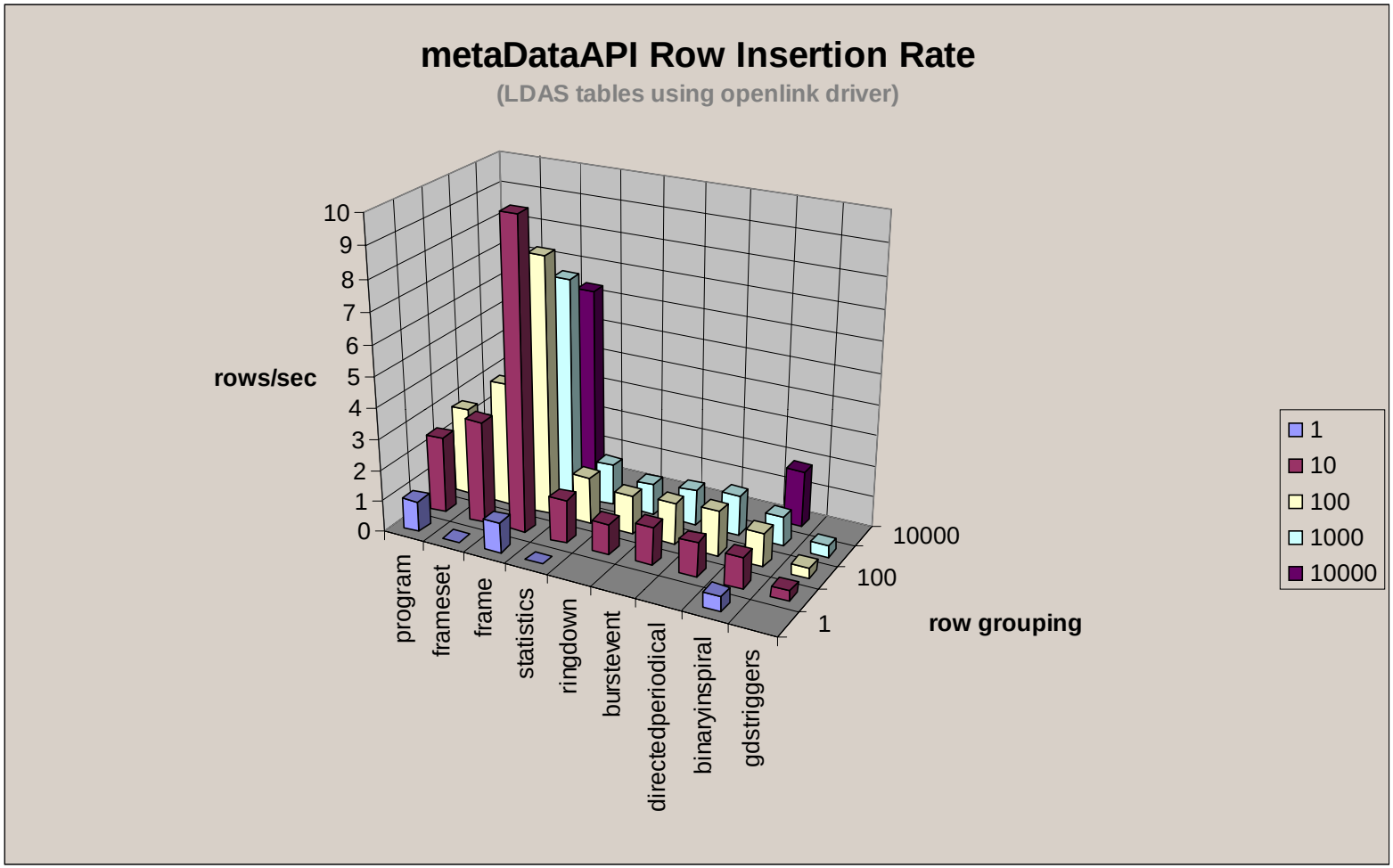


Linux Data Socket Tests

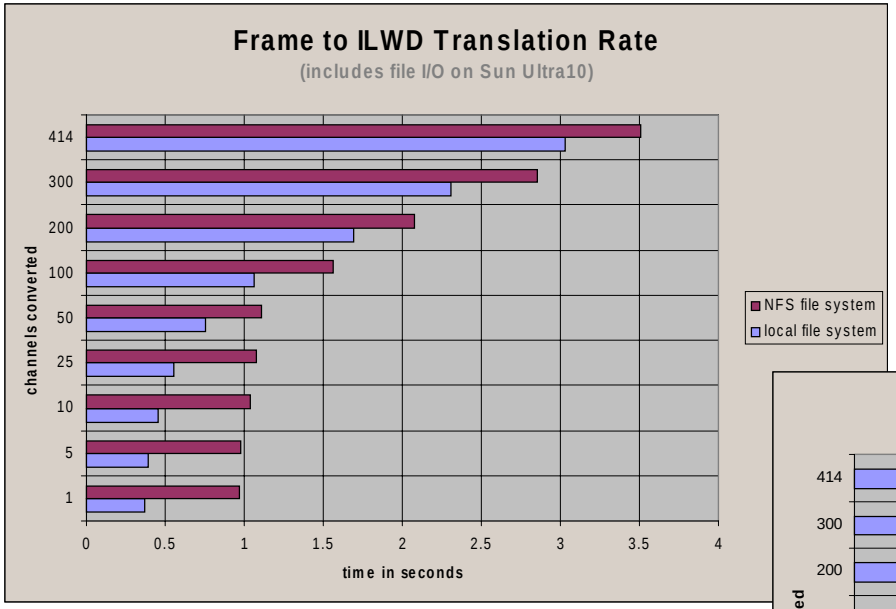


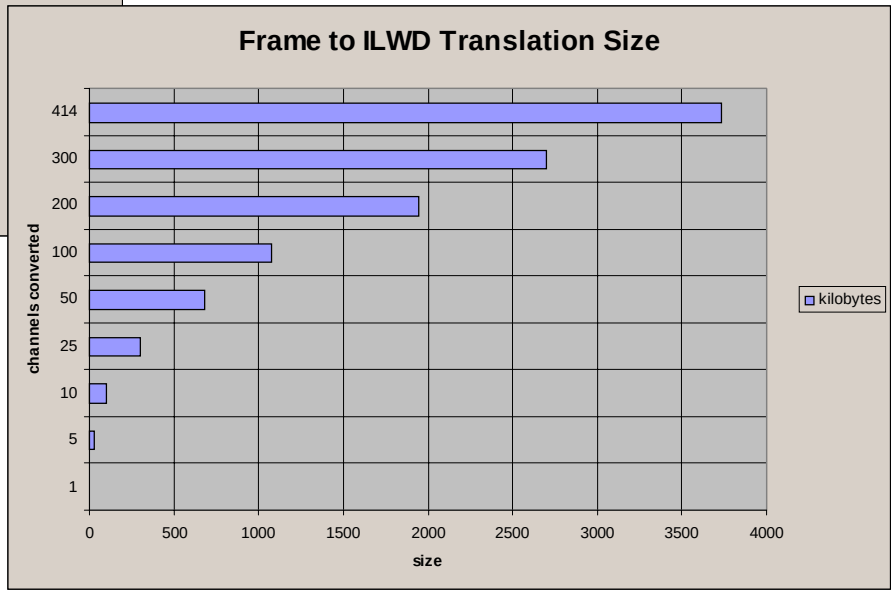


Metadata Ingestion Tests



Frame to ILWD Tests





Next Major Development

Parallel Computing Components

- mpiAPI:
  - detailed requirement published
  - to be written in TCL (no C++)
  - starts up mpirun script
  - maintains parallel queues
  - manages load balancing
  - communicates messages with the wrapperAPI
- wrapperAPI:
  - developing detailed requirements
  - is started by mpirun scripts
  - to be written in C++ (no TCL)
  - using MPICH 1.1.2
  - implementing ldas communicator class for load balancing
  - implementing generic algorithm class wrapper
  - master communicates with mpiAPI
  - slaves carry out parallel computing

Closing Remarks

- Even though measurable progress has been demonstrated, LDAS implementation has been very slow in starting up, requiring significant guidance over a young programming staff to implement the nearly complete LDAS software infrastructure.
- The modular design of the LDAS software will most likely facilitate more rapid code development once junior staff come up the learning curve and more complete software development documentation becomes available on the software developers' web pages.