How to Develop a LIGO Search

Peter Shawhan
(LIGO / Caltech)

NSF Review
November 18, 2003

Initial Exploration

I have an idea for a new search algorithm…

May use Matlab to refine it

- Quick development, built-in visualization, scripting

May want to look at some real data

- Tools for remote frame data retrieval: guild, getFrames
  Both require a valid LDAS username/password
- Matlab function to read from a frame file: frextract

Implementing the Algorithm

I'll write code in C

Link to LAL

- Standard data structures, input / output conventions
- Fourier transforms and other signal processing functions
- Error reporting and handling schema
- Does impose a certain programming style

I will want to structure my search code intelligently

- Separate algorithm itself from data input
- Put all algorithm parameters into a structure

May add enhancements to LAL package(s), or add new one

Can use existing code in CVS repository as examples

Testing and Tuning

Two approaches, depending on where I ultimately want to run the code:

Dynamic Shared Object (DSO) [*LALWrapper*]

- Designed to slot into LDAS
- Test / tune with "standalone wrapper" running on interactive machine
- Use LDAS to construct input data file (ilwd format)

Self-contained C program [*LALApps*]

- Use LAL support functions to read data, calibration, etc. from disk

May write to a verbose log file for debugging, dump intermediate products and view them with Matlab, etc.

Running the Search on Lots of Data

First, I have to know what data to analyze…

- Web site with lists of "segments", with data quality flags
- segwizard graphical interface, segments Tcl library

Run a DSO in LDAS

- Tool for remote job execution: ldasjob Tcl library
- Searches generally use a "loop script" written in Tcl
- Event candidates go into LDAS database
- Other output can go to disk files

Run a self-contained C program on a Condor cluster

- Again, need a script to define all the jobs with appropriate parameters
- Write a script to construct a "directed acyclic graph" (DAG)
- Output goes to disk files (*e.g.* event candidates in LIGO_LW format, using LAL functions)
- Grid computing is a natural extension of this approach

Statistical Analysis

Coincidence and statistical analysis

- Tools to retrieve event candidates from LDAS database: guild, getMeta
- For either environment, event candidates end up in LIGO_LW files
- Event lists can be read into Matlab (readMeta function), ROOT (eventTool extension), or a C program (metaio parsing library)

The details of the analysis depend on the search

- Coincidence requirements
- Vetoes, other cuts
- Background estimation

May do follow-up processing, *e.g.* cross-correlate raw data

- Full analysis “pipeline” might be quite complicated
- Goal is to automate as much as possible, focus on the high-level stuff