wrapperAPI: progress and recommendations

Patrick Brady: LIGO/LSC MPI working group

28 August 2000

1 Overview

The MPI working group was formed at the request of the LIGO laboratory and first met on February 24, 2000. The group includes Blackburn, Barnes and Lazzerini from the Laboratory, and Anderson, Brady, Brown, J Creighton, and T Creighton from UWM. Alan Wiseman has participated as a representative to the software committee.

The responsibilities of the working group as outlined in the first meeting were:

- To develop a fully functional LDAS wrapperAPI
- To demonstrate the concepts of the wrapperAPI and then later present it to a larger LSC community.
- To establish group communication utilities such as web page, minutes, as well as establish meeting guidelines (1 hour teleconference each week).
- To design target functionality for first dynamically loaded libraries which will be used in LSC/LIGO/LDAS mock-data-challenges.

A draft design of the wrapperAPI and mpiAPI were presented to the working group by Kent Blackburn at this first meeting.

After its first meeting, the group established a web page for communication of progress within the group [http://www.lsc-group.phys.uwm.edu/~patrick/ligo/mpi/].

2 Progress

2.1 Design review

The UWM group reviewed the preliminary wrapperAPI and mpiAPI design after the first meeting and made some recommendations for changes. (See notes on the web page.) The necessary changes were made to the wrapperAPI baseline requirements.

It was later realized that the proposed wrapperAPI design precluded the use of certain search codes without significant modification to the underlying algorithms. Hierarchical search code for binary
inspiral waveforms was identified as a particular example. Development was delayed by two weeks
to investigate this issue. The conclusions are detailed in a report on the web which explains that

1. the design of wrapperAPI might result in a loss of computational efficiency of the search code
   compared to the optimal “stand-alone” search code design. It was suggested that communi-
   cation be allowed inside the shared object functions to partially alleviate this problem.

2. It was also pointed out that dynamical load balancing would be very problematic within the
   optimal algorithm, and may be impossible for this type of search.

The conclusion drawn from this process:

any search envisioned by the UWM group could be made to fit into the wrapperAPI
baseline design by modifying the algorithm to be used in the search code. That is, the
algorithm used by any search code must be chosen from within the class of algorithms
implied by the wrapperAPI. It is anticipated that this may lead to reduced computational
efficiency of some search codes.

2.2 Code development

Code development for a prototype shared object library to fit into the wrapperAPI began in earnest
in mid April. It was decided that the UWM group would provide code to perform a flat (non-
hierarchical) binary inspiral search.

At this point, another issue was identified. The LAL uses a suite of purpose built error handling
structures and macros; they are particular to LAL. The five functions identified in the wrapperAPI
baseline requirements were not LAL compliant. A solution was proposed in and posted on the web
page. This called for:

1. conformance of functions within the shared object library to the LAL standard

2. inclusion in the wrapperAPI of the LAL error handling structure and macros

3. inclusion in the wrapperAPI of an additional function CheckLALStatusPtr() which would
   be executed after each call to a function in the shared object library.

This suggestion was rejected on the grounds that the wrapperAPI was not to depend explicitly on
code in LAL since this code was not under the control of the LDAS team.

The alternative solution, which was adopted, involved the design of an interface layer between the
functions defined in the wrapperAPI baseline requirements and LAL functions. These functions
would be written once (by Jolien Creighton) and used in all shared object libraries to provide the
interface between the wrapperAPI and the LAL.

Code was subsequently written to this design.

In June, Brady and Brown spent a month at LIGO (Caltech). During this time, Brown worked
closely with Barnes to understand the wrapperAPI code and its dependencies on other LDAS
components. The functions for the inspiral shared object were completed. And Brown spent a
considerable amount of time making LAL and inspiral search code available as shared objects to be linked into the wrapper at runtime. In mid-July, the prototype shared object library was available for compilation with the wrapperAPI, and testing began.

2.3 Testing

Preliminary testing of the wrapperAPI with the inspiral search code were successful.

To aid future development, it was decided to install LDAS on the UWM alpha linux Beowulf cluster. As of 26 August 2000, this was successfully achieved through the efforts of Barnes, Brown, and Charlton.

Since the preliminary tests of the wrapperAPI, further modifications have been made to the baseline requirements. These modifications broaden the class of algorithms that can fit into the current design.

The need for an additional function was identified to handle memory deallocation provided using LALMalloc() inside of applyFilters().

3 Current status

Since the formation of the working group, the wrapperAPI design has been revised to broaden the class of algorithms that can be used in search codes. The working group has:

1. developed a fully functional LDAS wrapperAPI
2. partially demonstrated the concepts
3. designed and implemented target functionality for first dynamically loaded library which will be used in LSC/LIGO/LDAS mock-data-challenges.

Thus, the working group has successfully completed the first stage of development and testing of the wrapperAPI and the LDAS parallel computing infrastructure.

4 Conclusions and recommendations

The current design of the wrapperAPI has positive and negative features. Positive features which should be preserved in future designs include:

- The notion of a generic interface between LDAS and LAL search codes.
- The implementation using shared object libraries which can be dynamically loaded.

Negative features of the current wrapperAPI design include;

- the implied (class of) algorithms implicit in the current wrapperAPI.
• the requirement that the wrapperAPI should not depend on LAL code for compilation.
  
  – the wrapperAPI should use the error reporting and memory handling facilities provided in LAL

It is the considered opinion of the working group that the wrapperAPI design should be modified so that it equally reflects the need for control/management within LDAS and the need for efficient and flexible science code. A design review is advisable at this time. Once a design is settled, a minimal delay (of order 2 weeks) can be expected in order to implement the modified wrapperAPI design.

A draft document which revisits the requirements of the LDAS parallel computing facilities and which provides specifications for a modified wrapperAPI and specifications for LAL search codes is in active preparation and a draft is available on the web page.