LDAS-LAL Interface (*wrapperAPI*)

LSC Meeting
August 15-17, 2000
LIGO Hanford Observatory

James Kent Blackburn
California Institute of Technology

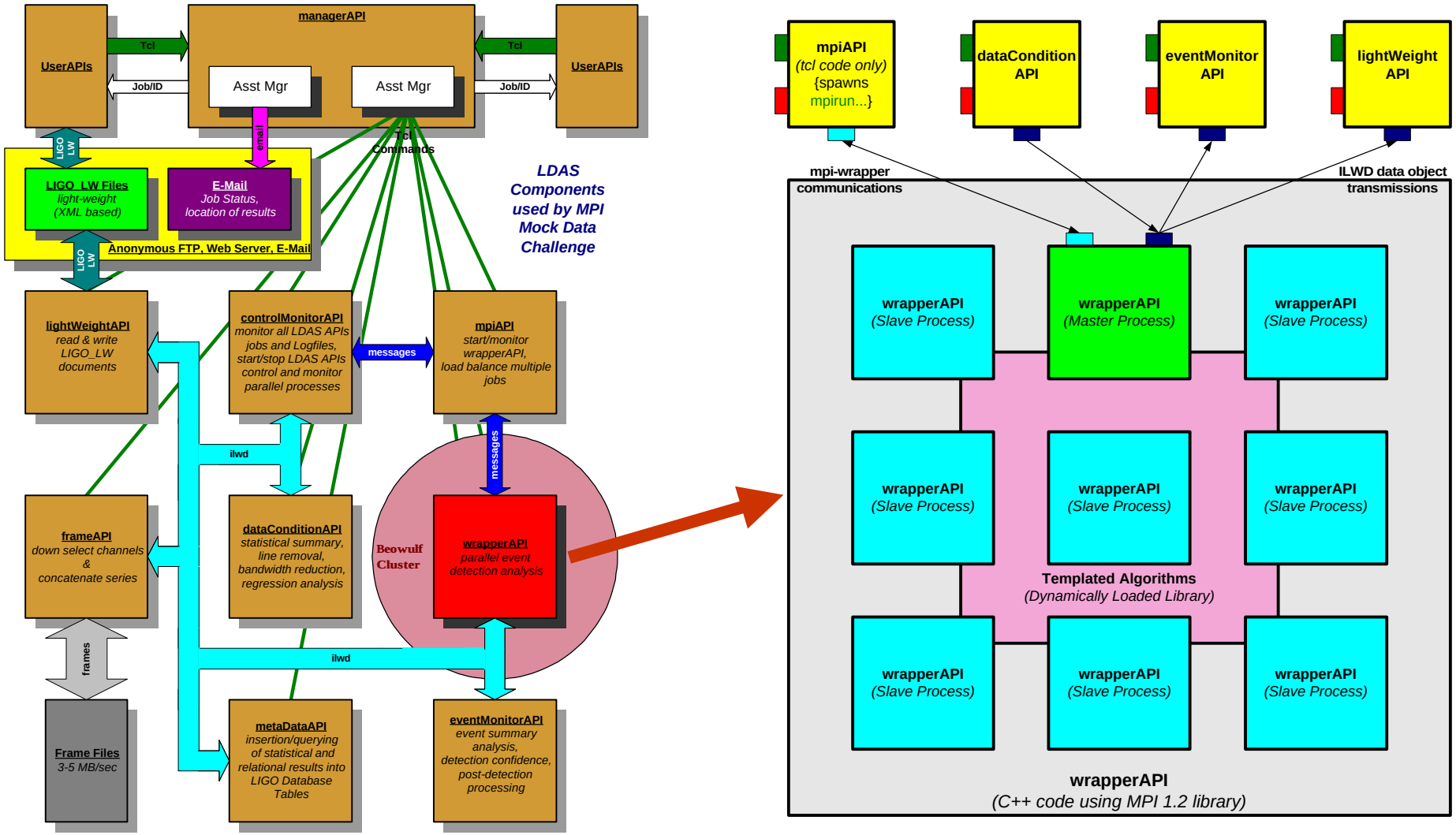
LIGO-G000200-00-E

August 16, 2000

wrapperAPI highlights

- Parallel computing component of the LDAS system.
- Bridge between LDAS system software and LAL algorithmic search software.
- Two parts, LDAS API + dynamically loaded shared object library (dso) containing LIGO search engine.
- Decouples LDAS builds from LAL builds.
- Isolates search development teams from learning LDAS specific coding standards.
- Integrated into LDAS health and status environment.
- Based on MPI using MPI-2 C++ API from Notre Dame (supports MPICH and LAM).
- Shared objects written in C following LAL specifications.

block diagram details



parallel architecture

- wrapperAPI manages parallel analysis flow
- Only the master process communicates directly with LDAS
- Not necessary for LSC developers to master MPI to develop search algorithms.
- However, LAL search algorithms may use MPI based parallel analysis flow control.
- Nominally data communications between master and slave processes handled by wrapper, though communication of input data to slaved can be deferred to the search algorithm via the conditionData() interface (Note: MPI needed here).

complex communication

- Five unique interfaces built into wrapperAPI:
  - Command line arguments for initialization of search algorithms
  - LDAS command/message interface
  - LDAS ILWD data communications
  - MPI standard parallel message passing
  - Six C level interface functions interact with dynamically loaded shared objects (LAL code).

command line options

mpirun wrapperAPI …

- **-np N**
- **-machinefile /full/path/machine.in**
- **-nolocal**
- ***other mpirun options not generally used***
- **-nodelist = {i-j,k,l,m-n,…}**
- **-dynlib = /full/path/libname.so**
- **-mpiAPIport = {hostname, socketport}**
- **-dataAPI = {hostname, socketport}**
- **-resultAPI = {hostname, socketport}**
- **-dataDistributor = {WRAPPER||CONDITIONDATA}**
- **-nodeDutyCycle = N**
- **-slaveReportCycle = N**
- **-communicationOutput = {ALWAYS||ONCE}**
- **-filterParams = {a,b,c,d,…}**
- **-realTimeTatio = n.mmmmmm**
- **-doLoadBalance = {TRUE || FALSE}**

LDAS commands/messages

- wrapperAPI to mpiAPI:
  - #:request add N
  - #:request sub N
  - #:warning {list of warnings}
  - #:error {list of errors}
  - #:progress nnn.mmm%
  - #:using N {i-j,k,l,m-n,…}
  - #:projected ratio n.mmmmm
- mpiAPI to wrapperAPI:
  - #:add N {i-j,k,l,m-n}
  - #:sub N {I-j,k,l,m-n}
  - #:kill
  - #:cont

C interface functions

- Current definition targets efficient/rapid parallel implementation:

```
INT4 initFilters( INT4 argc, CHAR* argv[], CHAR** initMessage );
INT4 indexFilters( UINT4* numberFilters, CHAR** indexMessage );
INT4 conditionData( inPut* data, CHAR** conditionMessage, MPI_Comm* comm );
     do {
INT4 templateFilters( INT4 beginIndex, INT4 endIndex, const inPut* data,
     outPut** results, CHAR** filterMessages );
INT4 freeOutput( outPut** results, CHAR** freeMessage );
     } while (!finished);
INT4 freeFilters( CHAR** freeMessage );
```

- C functions are called by the wrapperAPI according to the above schedule.
- Exact details of flow can be customized by several of the command line options
- At the LAL level exists a “dual” set of these functions which follows the LAL standards and is only written once.

wrapperAPI status

- Draft of “The wrapperAPI’s baseline requirements”, LIGO-T990097-09-E available on web.
- wrapperAPI level approximately 90% complete per LIGO-T990097-09.
- MPI working group involving LDAS and UWM formed in February of this year.
- UWM has developed a dynamically loaded shared object for testing which was successfully run on the LDAS beowulf in July of this year.
- MPI data communication performance has been increased in the wrapperAPI level by 10x since earliest runs.

Closing Remarks

- Best utilization of the wrapperAPI requires that dso's be written such that all slave processes are working in parallel (distributed model supported but does not have access to full suite of load balancing and runtime tuning features of the wrapperAPI).
- Communications overhead has been tested and greatly improved on hardware limited to 16 CPUs running at 333MHz. LIGO I Science run will probably target order 100 CPUs running at ~1000MHz, an environment far more taxing on communication overheads – new command line options have been added to the wrapperAPI to remove most overhead which must be used in conjunction with constraints on result data produced by dso's.
- The C++ MPI 2 API that is being used has been tested only with MPICH version 1.2. LAM is also documented as being supported but is untested. Lam is now shipped with Redhat and development has been moved from Ohio State to Notre Dame.
- wrapperAPI needs missing LDAS components (mpiAPI, controlMonitorAPI, eventMonitorAPI) to fully integrate.
- Full scale planning for December/January MDC should begin immediately.