Current Status of KAGRA data analysis library, KAGALI

7th Japan-Korea Meeting
December 20, 2014
@Toyama Univ.

Hirotaka Yuzurihara
Osaka City University
KAGRA Data analysis (CBC search)
What is KAGALI?

- KAGRA data analysis subsystem develops our own data analysis library called KAGra Algorithmic Library or “KAGALI” in short.

- LSC software, especially LAL, is likely to put a higher priority to safety in coding than to speed in execution. We will try to improve execution performance keeping mechanisms for easily finding and removing bugs as well.

- We should prepare software proper to KAGRA including data handling tools for ourselves.
Coding style guide of KAGALI for C and C++

- Coding style guide by Y. Itoh (for C) and by H. Takahashi (for C++).
- You can read the latest version:
guide line = ancestors’ wisdoms

(Itoh-san’s slide)

• Do not make bugs sneak into your code.
  – Use techniques which has been well-tested and sophisticated in the past.

• Make a code re-useable.
  – Make your code as easily available as possible to even people who read only the API manual of your code, but not the source.

• Make your code readable: If it is, we (including you in near future) can easily debug it, review it, and extend its functionality, thereby increase its credibility.
External Library using KAGALI

KAGALI requires the installation of the external libraries below.

- **FFTW3**  [http://www.fftw.org/](http://www.fftw.org/)
  computing the discrete Fourier transform in Fast Fourier Transform (FFT)

  provide various mathematical routines
  for example random number generators, special functions and probability
  density function of several distributions.

- **Frame Library**  [http://lappweb.in2p3.fr/virgo/FrameL/](http://lappweb.in2p3.fr/virgo/FrameL/)
  dedicate to frame manipulation including file input/output
  usually outputs from GW detectors and information of detector are filled
  in frame format

KAGALI can compile and work at least the below version,

- **FFTW 3.3.4**
- **GSL 1.16**
- **Frame Library 8.20**
Manage of KAGALI

- Git server controls KAGALI version.
  Each executable program contains information of git version for debugging.

- When you copy the latest repository (password protected),
  $ git config --global http.sslVerify false  <- set up around SSL
  $ git clone https://vt001.resceu.s.u-tokyo.ac.jp/git/kagali-v0r1a
  $ cd kagali-v0r1a
  $ make; make install

    -> static library(*.a) will be generated in $(PREFIX)/kagali/lib/
    header file(*.h) will be generated in $(PREFIX)/kagali/include/kagali/

    (default : PREFIX=$HOME)

- autoconf / automake will be developed in future.

- We are developing KAGALI just now for data taking of iKAGRA (Dec. 2015).
  Co-developer and user of KAGALI are welcome!
Structure of KAGALI

mainly two directories

- **kglcommon**
  - common function in any pipeline
  - for example, frame file I/O, FFT, error handling, physical constant, ..

- **cbc**
  - specific function for CBC(compact binary coalescence) search
  - for example, generate template bank, matched filter, chi square veto, ..

-> We are developing offline and online search pipeline.
Examples of KAGALI function

- Inverse Fast Fourier Transform

```c
void KGLReverseRealFFT( //begin{proto}
    KGLStatus     *status, /**< status */
    KGLRealFFTPlan **plan, /**< (in/out) KGLRealFFTPlan will be stored in *plan */
    double         *rdata, /**< (out) returns the reverse Fourier transform */
    const double complex *cdata, /**< (in) gives a complex data */
    const size_t     size   /**< (in) gives the size of the real data */
) //end{proto}
```

- All KAGALI function have status pointer. (KGLstatus)
  Via this status pointer, error handling of each functions is done.

- If error happened,
  - `status-> statusCode` stores error code (KGL_EL_ERROR, KGL_EL_WARNING, ..)
  - `status-> message` stores error message
    (“size must be larger than 0\n”)
Each function in KAGALI check arguments at top of function using KGLAssert, which is an expression verification macro.

The second arguments is a logical expression which should be satisfied.

```c
KGLAssert(status,cdata != NULL,"cdata is NULL");
KGLAssert(status,rdata != NULL,"rdata is NULL");
KGLAssert(status.size > 0,"size must be larger than 0");
if(KGLCheckError(status)) return;
/* end of check on parameters */
```

If expression is true, KGLAssert macro does nothing. If not, status->statusCode is set to be KGL_EL_ERROR and the message string given as the third argument is stored status->macro.

Example:

```
gcm stdin 0 FFT.fft.0 ggb2 matlab 02 E:
KGL Error: Assertion: size must be larger than 0
in function KGLForwardRealFFT (line 277 of RealFFT.c)
KGL Abort in function main (line 96 of fft.c)
Abort
```

Example:
Summary of developed functions in KAGALI

a part of functions which we have developed

- fft : (Ohara-san)
  - KGLRealForwardRealFFT
  - KGLRealReverseRealFFT
  - KGLAddRealFFTPlan
  - RealDestoryRealFFTPlan
  - KGLRealForwardRealFFTOnce
  - KGLRealReverseRealFFTOnce
  ...

root/kglccomon/
Summary of developed functions in KAGALI

- `filter` : (Hayama-san, Ueno-san)
  - KGLFIRFilter
  - KGLIIRFilter
  - KGLWindowFunction
  - KGLLowPassFilterKernel
  - KGLHighPassFilterKernel ..

- `PSDestimate` : (Yuzurihara)
  - KGLAveragePowerSpectrum
Summary of developed functions in KAGALI

- **frameIO**: (Yuzurhara)
  - KGLReadFrame2ProcData
  - KGLProcData2WriteFrame ..

- **detector noise**: (Ueno-san)
  - KGLReadNoisePSD ..

- **interpolation**: (Ueno-san)
  - KGLLinearInterpolation
  - KGLChebyshevPolation ..

- ...

We are developing KAGALI for iKAGRA (Dec. 2015).
KAGRA data analysis subsystem develops our own data analysis library called KAGra Algorithmic LIbrary or “KAGALI”.

KAGALI coding style guide has been prepared by Itoh-san and Takahashi-san.

We have developed basic components of KAGALI. These include a wrapper for the FFTW library, the wrapper for the Frame Library to read Frame format data, and other function (see previous slide).

CBC offline and online search pipeline is in progress.

We are developing KAGALI for iKAGRA observation (Dec. 2015).