Geodetic First Approximation of Size and Timing: User's Guide

Brendan Crowell [*1] and Ben Baker [†2]

[1]Pacific Northwest Seismic Network
[2]Instrumental Software Technologies, Inc.

April 30, 2018

Contents

*crowellb@pnsn.org
†benbaker@isti.com

1 Introduction

copy brendan's original manual, note the amazon funding, and add brendan's publications so people know what to cite

G-FAST (Geodetic First Approximation of Size and Timing) was developed by Brendan Crowell as a Python based geodetic earthquake early warning (EEW) module. Funding from Amazon has allowed PNSN to developed a compiled-language production variant of G-FAST for integration into the USGS's next generation ShakeAlert EEW system. The software is released under the license described in Appendix A.

Nominally, G-FAST acquires real-time geodetic data and, when triggered by a ShakeAlert message, activates three geodetic co-seismic inversion suites: a peak ground displacement (PGD) inversion, a centroid moment tensor (CMT) inversion, and a CMT driven slip or finite fault inversion. The PGD module provides a quick and robust estimate of magnitude. The CMT inversion provides a robust estimate of magnitude, depth, and moment tensor components. The moment-tensor derived fault planes and depth are then passed onto the slip inversion which resolves the CMT fault plane ambiguity while providing an additional magnitude estimate. The PGD and slip model are then passed onto the ShakeAlert decision module as the event unfolds.

As in the Python implementation, G-FAST still acquires real-time data from the Pacific Northwest Geodetic Array (PANGA) and, in the EEW context, is triggered on ShakeAlert messages. The key differences are now that an Earthworm server is responsible for the import of PANGA data and the ShakeAlert communication is directly implemented via ActiveMQ. Additionally, G-FAST has been separated into multiple C-libraries to encourage reuse by groups outside of EEW.

2 Installation

This section outlines the GFAST installation procedure. This consists of two primary steps. The first step is to install the dependencies. The second step is to then build GFAST and link to the dependencies.

2.1 Prerequisites

This section lists the requisite, optional, and workaround libraries for building GFAST.

2.1.1 Compilers

The following libraries and GFAST can be compiled with a C, C++, and Fortran compiler. It is recommended the compilers have complete OpenMP-4 support for improved code vectorization. OpenMP-4 support is realized in GCCv5 and above or Clang 3.9 and above though GFAST has been successfully built with GCCv4.4. If using GCCv4 then it is recommend one set the "-Wno-unknown-pragmas" C compiler flag.

2.1.2 CMake

CMake is used because G-FAST is decomposed into many smaller libraries with varying dependencies. This dependency ambiguity is implicitly handled by this automatic makefile generator. In addition to generation of makefiles also CMake provides for automatic generation and execution of unit tests. As with any makefile creation step the correct specification of dependencies can be quite painful but, upon successful generation of a CMakeCache.txt, file this activity need not be repeated and subsequent pulls from the git repository should build without hassle. CMake is likely available through a package manager or can be obtained from `https://cmake.org/`. GFAST has been successfully built with Version 2.6 and above.

2.1.3 LAPACK and BLAS

G-FAST ultimately aims to solve matrix-vector equations of the form $\|G\mathbf{m} = \mathbf{d}\|_2$ (LAPACK) or estimate data by computing matrix-vector multiplies of the form $\mathbf{u} = G\mathbf{m}$ (BLAS). Moreover, profiling shows that computation of the least-squares problem, particularly in the finite fault inversion, is the greatest contributor to program execution time. Necessarily, one can easily achieve better performance by using high-performance vendor LAPACK and BLAS library implementations. Consequently, it is recommended one follow the strategy of

1. Check for an existing vendor LAPACK/BLAS implementation.

2. If using an Intel processor obtain Intel's Math Kernel Library (MKL) which is freely available at `https://software.intel.com/sites/campaigns/nest/`. Also, note that MKL contains an implementation of FFTw and may reduce the number of dependencies.

3. Check a package manager for a pre-built LAPACK and BLAS which are distributed with a modified BSD license.

4. TODO: look for OpenBLAS then, if not found, ATLAS in the CMakeModules if using a package manager's lapacke + blas.

5. At last resort obtain and build LAPACK and BLAS from `http://www.netlib.org/lapack/` which are distributed with a modified BSD license.

2.1.4 libGeographic

This is a library used by GFAST's coordinate tools unit tests. It may be available through a package manager or, if need be, obtained from `http://geographiclib.sourceforge.net/`. It is distributed under the MIT license.

2.1.5 iniparser

This is the utility which parses the GFAST parameter file. It is available from `https://github.com/ndevilla/iniparser` and is distributed under the MIT license.

2.1.6 HDF5

The HDF5 self-describing file format is used for high-performance and portable data archival and data playback. HDF5 can be obtained from a package manager or from `https://www.hdfgroup.org/HDF5/`. GFAST appears to work with HDF5 1.10.1. If static linking to HDF5 then one must additionally obtain the zlib compression library. A zlib implementation is available in Intel's Performance Primitives available at `https://software.intel.com/sites/campaigns/nest/`. Zlib can also be obtained with a package manager or from `http://www.zlib.net`. Notice that there is no data compression in GFAST disk writes and this step can be avoided by linking to the shared HDF5 library. If choosing between IPP and a zlib then note that IPP improves the performance of ISCL. Zlib and HDF5 both are distributed with permissive software licenses.

2.1.7 libxml2

The ShakeAlert and QuakeML data products are created with libxml2. If using GCC then it is very likely you already have libxml2. Otherwise, it can be obtained from `http://xmlsoft.org/`. libxml2 is distributed under the MIT license.

2.1.8 ISCL

Brendan's initial GFAST implementation was written in Python and made extensive use of Numerical Python. Somewhat fortuitously ISTI had been developing a library targeted at expediting the conversion of NumPy laden Python scripts to C via the ISTI Scientific Computing Library (ISCL). ISCL is distributed under the Apache-2 license and freely available from `https://gitlab.isti.com/bbaker/iscl`.

2.1.9 compearth

This is a moment tensor manipulation package developed by Carl Tape and recently ported to C by ISTI. It is used by G-FAST to perform moment tensor decompositions. It is currently available from `https://github.com/bakerb845/compearth` and the C source exists in compearth/momenttensor/c_src. It may soon be merged back into Carl's original directory. It is distributed under the MIT license.

2.1.10 ActiveMQ

The ShakeAlert earthquake early warning system sends and receives alerts with the ActiveMQ messaging system. If one is not using GFAST in earthquake early warning then ActiveMQ is not necessary. ActiveMQ additionally depends on libcrypto and libssl. ActiveMQ is distributed under the Apache 2 license is available at `http://activemq.apache.org/cms/download.html`.

2.1.11 Earthworm

The real-time GPS data are currently incorporated into GFAST via Earthworm which is available at `http://earthworm.isti.com/trac/earthworm/`. If not using the real-time system then Earthworm is not necessary. Earthworm requires two additional libraries, RabbitMQ-c and Jansson for import of the GeoJSON which contains the GPS precise-point-position data. RabbitMQ-c is available at `https://github.com/alanxz/rabbitmq-c` and is distributed with an MIT license. Jansson is available at `https://github.com/akheron/jansson` and is distributed with a permissive license.

2.1.12 FFTw

The requirement for this library will be dictated by your linking policy. If you require static linking *and* are not using Intel MKL then to successfully build GFAST you would have to obtain FFTw from a package manager or obtain it from `http://www.fftw.org/`. Notice however that no Fourier transforms are computed in GFAST. Additionally, if static linking without MKL then FFTw's GPL-2 copyleft will impact GFAST's internal license. To avoid this complication it is recommended one use MKL or link to the dynamic ISCL library.

2.1.13 Cython

A developmental Python interface is provided through Cython which converts Python to C and is available at `http://cython.org/` or through a package manager. This is not required and at the present time minimally supported.

2.2 Building G-FAST

To increase the likelihood that G-FAST can be compiled on multiple platforms (e.g. Linux, Windows, OSX) and ease the burden on developers to modify cumbersome makefiles G-FAST adopts the CMake automatic makefile generation utility. Naively, one can descend into G-FAST's root directory and type

```
cmake .
```

The CMakeModules will attempt to find the requisite libraries and may be moderately successful if items are installed in 'standard' locations.

Alternatively, if one would prefers a less serendipitous build then the paths to the custom directories can be provided directly to cmake

```
cmake ./ -DCMAKE_BUILD_TYPE=DEBUG \
      -DCMAKE_INSTALL_PREFIX=./ \
      -DCMAKE_C_FLAGS="-g3 -O2 -fopenmp -Wall -Wno-unknown-pragmas" \
      -DEW_BUILD_FLAGS="-Dlinux -D_LINUX -D_INTEL -D_USE_SCHED  -D_USE_PTHREADS" \
      -DCMAKE_CXX_FLAGS="-g3 -O2" \
      -DGFAST_INSTANCE="YourInstitution" \
      -DGFAST_USE_AMQ=TRUE \
      -DGFAST_USE_EW=TRUE \
      -DAPR_INCLUDE_DIR=path_to_include_apr-1 \
      -DLIBAMQ_INCLUDE_DIR=path_to_include_activemq-cpp-3.8.2 \
      -DLIBAMQ_LIBRARY=path_to_libactivemq-cpp \
      -DLSSL_LIBRARY=path_to_libssl \
      -DLCRYPTO_LIBRARY=path_to_libcrypto \
      -DLAPACKE_INCLUDE_DIR=path_to_lapacke_include \
      -DLAPACKE_LIBRARY=path_to_liblapacke \
      -DLAPACK_LIBRARY=path_to_liblapack \
      -DCBLAS_INCLUDE_DIR=path_to_cblas_include \
      -DCBLAS_LIBRARY=path_to_libcblas \
      -DBLAS_LIBRARY=path_to_libblas \
      -DH5_C_INCLUDE_DIR=path_to_hdf5_include \
      -DH5_LIBRARY=path_to_libhdf5 \
      -DINIPARSER_INCLUDE_DIR=path_to_iniparser_src \
      -DINIPARSER_LIBRARY=path_to_libiniparser \
      -DCOMPEARTH_INCLUDE_DIR=/path_to_compearth_include \
      -DCOMPEARTH_LIBRARY=/path_to_libcompearth \
      -DISCL_INCLUDE_DIR=path_to_iscl_include \
      -DISCL_LIBRARY=path_to_libiscl_static \
      -DGEOLIB_LIBRARY=path_to_libGeographic \
      -DFFTW3_LIBRARY=path_to_libfftw3 \
      -DEW_INCLUDE_DIR=path_to_earthworm_include \
      -DEW_LIBRARY=path_to_earthworm_libew \
      -DLIBXML2_INCLUDE_DIR=path_to_libxml2 \
      -DLIBXML2_LIBRARY=path_to_libxml2
```

As another example, one can use the MKL and IPP libraries to boost performance

```
cmake ./ -DCMAKE_BUILD_TYPE=DEBUG \
      -DCMAKE_INSTALL_PREFIX=./ \
      -DCMAKE_C_COMPILER=/usr/bin/gcc \
```

```
-DCMAKE_CXX_COMPILER=/usr/bin/c++ \
-DCMAKE_C_FLAGS="-g3 -O2 -fopenmp -Wall" \
-DEW_BUILD_FLAGS="-Dlinux -D_LINUX -D_INTEL -D_USE_SCHED  -D_USE_PTHREADS" \
-DCMAKE_CXX_FLAGS="-g3 -O2" \
-DGFAST_INSTANCE="YourInstitution" \
-DGFAST_USE_INTEL=TRUE \
-DGFAST_USE_AMQ=TRUE \
-DGFAST_USE_EW=TRUE \
-DAPR_INCLUDE_DIR=path_to_apr \
-DLIBAMQ_LIBRARY=path_to_libactivemq-cpp \
-DLSSL_LIBRARY=path_to_libssl \
-DLCRYPTO_LIBRARY=path_to_libcrypto \
-DMKL_LIBRARY="mkl;libraries;" \
-DIPP_LIBRARY="intel;performance;libraries" \
-DH5_C_INCLUDE_DIR=path_to_hdf5_include \
-DH5_LIBRARY=path_to_lib_libhdf5 \
-DINIPARSER_INCLUDE_DIR=path_to_iniparser_src \
-DINIPARSER_LIBRARY=path_to_libiniparser \
-DCOMPEARTH_INCLUDE_DIR=/path_to_compearth_include \
-DCOMPEARTH_LIBRARY=/path_to_libcompearth \
-DISCL_INCLUDE_DIR=path_to_iscl_include \
-DISCL_LIBRARY=path_to_libiscl \
-DGEOLIB_LIBRARY=path_to_libGeographic \
-DEW_INCLUDE_DIR=path_to_ewinclude \
-DEW_LIBRARY=path_to_libew \
-DLIBXML2_INCLUDE_DIR=path_to_libxml2_include \
-DLIBXML2_LIBRARY=path_to_libxml2
```

In both instances the Earthworm pre-processor flags were obtained from the appropriate $GLOBALVARS$ as to be consistent with Earthworm headers.

Note, the above examples provide the option of building with ActiveMQ and Earthworm. This behavior is controlled through the DUSE_AMQ and DUSE_EW flags respectively. If either flag is not set then the default behavior should be to false. This may be advantageous when building GFAST in a pure playback mode of operation. In this case one would not need to specify the corresponding ActiveMQ and Earthworm library and include directories.

2.3 Generating the Doxygen Documentation

Doxygen is a useful tool for generating a type of code documentation. It is particularly useful to developers who wish to look-up variables in structures, variable meanings in function calls, and program function dependencies. Doxygen is unobtrusively used by decoarting preambles in source files and structures in header files. After running doxygen on the Doxyfile in the root source directory, Doxygen will scan the predetermined source folders and autogenerate a large amount of documentation. The output can be viewed with a web-browser or PDF viewer.

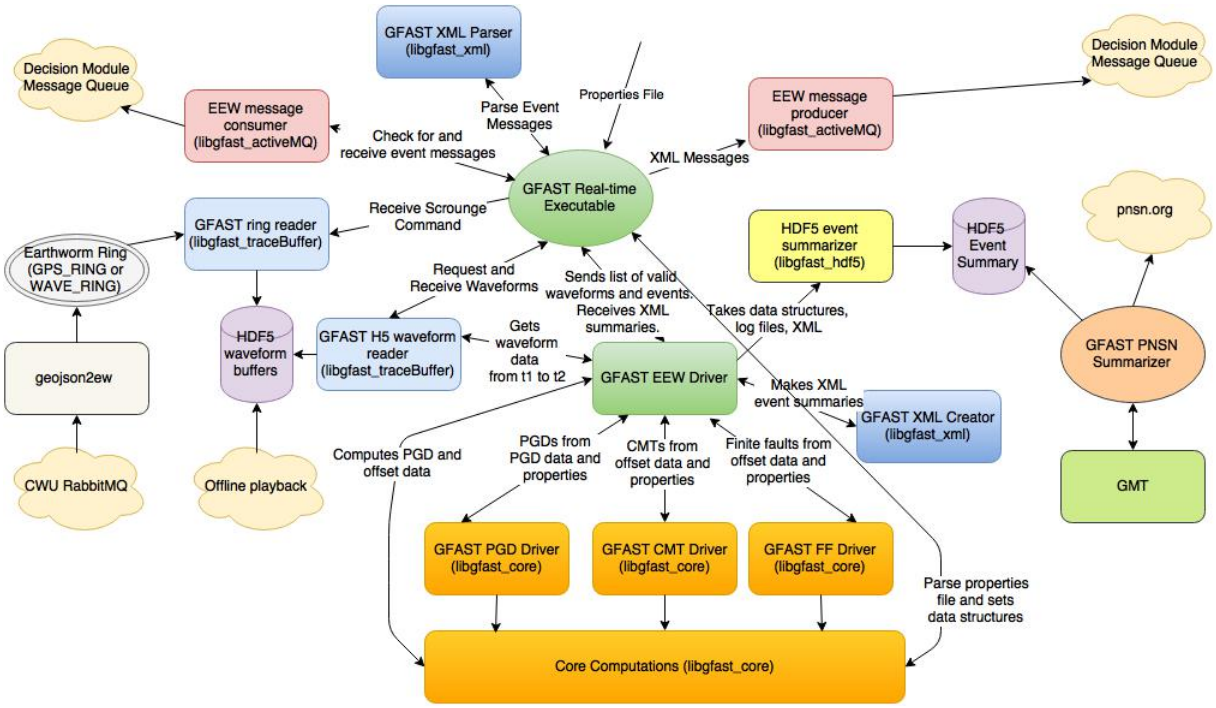

Figure 1: Overview of G-FAST and its libraries.

3 G-FAST Library Overview

This section will outline the G-FAST libraries as summarized in Figure 3. Libraries are used extensively to emphasize modularity, allow users to customize G-FAST to their application without incurring the cruft of another G-FAST application, and promote a maintainable software framework whereby software fixes can efficiently propagate to all users. The libraries to be introduced are

- Subsection 3.1: libgfast_core for core G-FAST modeling and inversion
- Subsection 3.2: libgfast_activeMQ for G-FAST communication in ShakeAlert
- Subsection 3.3: libgfast_hdf5 for G-FAST archival of heavy data
- Subsection 3.4: libgfast_xml for parsing and generating QuakeML and ShakeAlert XML
- Subsection 3.5: libgfast_traceBuffer for data IO from Earthworm and HDF5
- Subsection 3.6: libcmopad for moment tensor decompositions

3.1 G-FAST Core Routines

This section outlines the libgfast_core utilities which are essential to the expert earthquake-early warning driver routines. The six primary utilities are

- A properties reader for requisite G-FAST modeling parameters.
- Waveform processing for computation of peak ground displacement or offsets derived from precise-point-position time series data.
- PGD inversion of peak ground displacement data computed in the waveform processing.
- CMT inversion of offset data computed in the waveform processing.
- Finite fault inversion of data computed in the waveform processing.
- Coordinate utilities for converting latitude and longitude to and from UTMs[1].
- An event list handler for handling multiple events.

3.1.1 G-FAST Properties Reader

The G-FAST properties directory is responsible for reading all properties from the G-FAST properties file. To mimic the functionality of G-FAST will have general properties, PGD scaling properties, CMT inversion properties, finite-fault inversion properties. and, optionally, ActiveMQ properties and Earthworm properties. The properties will be discussed at greater length in Section . In this directory also exists a convenience routine for clearing the G-FAST properties during a finalization phase as well as displaying the properties to a text file or standard out.

3.1.2 G-FAST PGD Scaling

The G-FAST peak ground displacement inversion accepts from the waveform processor the largest[2] peak ground displacements in a valid region around the earthquake hypocenter. This valid region is computed from a configurable S-wave masking velocity to prevent offsets at different stations from contaminating the inversion. From this PGD data, G-FAST will

$$gm = u \tag{1}$$

3.1.3 G-FAST CMT Inversion

The G-FAST CMT inversion accepts from the waveform processor average static offsets for each station and component from the time a theoretical S-wave has passed through the station to the latest available data point. Like in the PGD scaling the CMT S-wave mask velocity is configurable. Provided the theoretical S-wave has passed through a configurable

[1]This directory supersedes the libGeographic ISCL interface as libGeographic has unpredictable behavior when crossing UTM zones that could jeopardize program library execution in the Pacific Northwest.

[2]Largest is maximum observed Euclidean distance of the three-component offset data within a valid S-wave mask.

minimum number of stations the inversion will perform for each depth a linearized moment tensor inversion. The moment tensor inversion is constrained to purely deviatoric sources for stability.

$$G\hat{\mathbf{m}} = \mathbf{d} \tag{2}$$

where

$$\hat{\mathbf{m}} = \begin{Bmatrix} m_{xy} \\ m_{xz} \\ m_{zz} \\ \frac{m_{xx}-m_{yy}}{2} \\ m_{yz} \end{Bmatrix} \tag{3}$$

and

$$G = \tag{4}$$

As a word of caution, observe that the output moment tensors have been converted to a more conventional North, East, Down (e.g. **?**) $\{m_{xx}, m_{yy}, m_{zz}, m_{xy}, m_{xz}, m_{yz}\}^T$ for the HDF5 output and Up, South, East (e.g. **?**) $\{m_{rr}, m_{\theta\theta}, m_{\phi\phi}, m_{r\theta}, m_{r\phi}, m_{\theta\phi}\}^T$ for the QuakeML output.

After the grid-search is complete a double-couple promoting penalty is applied at each depth, $i = 1, 2, \cdots, n_d$, to the variance reduction, $v_r^{(i)}$

$$\hat{v_r}^{(i)} = \frac{v_r^{(i)}}{\mathrm{DCP}\{\mathbf{m}^{(i)}\}} \tag{5}$$

where DCP denotes the double couple percentage of the moment tensor.

3.1.4 G-FAST Finite Fault Inversion

3.1.5 G-FAST Event Handler

In real-time applications it is possible multiple events can occur within a modeling window. To accomodate this G-FAST has a small facility for event bookkeeping that relies on the assumption that all events have unique event IDs. These functions can add a new event to the processing list or remove an event if it has surpassed its maximum modeling time (expired), or remove an event if its event ID was cancelled.

3.2 G-FAST ActiveMQ Messaging

The ShakeAlert framework requires a flexible message service to allow different modules to communicate. This is realized with the ActiveMQ library. G-FAST makes very limited use of ActiveMQ for EEW ShakeAlert compliance. G-FAST can subscribe to a message queue whereby it consumes XML messages from the decision module. G-FAST can also subscribe to a message queue to which it produces XML messages for future ShakeAlert use. To use both features one mess correctly enter the credentials specified in the properties file.

3.3 G-FAST HDF5 Archive

Presently there exists a framework for creation and dissemination of ShakeAlert products via XML. However, as an algorithm's products and complexity increasses it becomes important to consider frameworks for saving the heavy data in the applications highspeed memory to disk. This is accomplished in a portable and flexible way with self-describing binary HDF5 file format. Using HDF5 can copy then write to disk each application data structure for every G-FAST iteration.

3.4 G-FAST XML Creation and Parsing

The eXtensible Markup Language format is a portable human and machine readable file format used by ShakeAlert for communication of core data products (e.g., hypocenter, origin time, magnitude). G-FAST and the other geodetic algorithms will extend the core products to include magnitudes from PGD inversion and fault slip inversions. Additionally, because G-FAST computes CMTs, G-FAST will offer an XML based earthQuake Markup Language (quakeML) CMT summary which may be of use to NEIC earthquake contributors.

3.5 G-FAST Trace Buffers

G-FAST is intended for use in both real-time and purely playback scenarios with the underlying assumption that it was not to be an Earthworm module. To achieve this flexibility a common HDF5 data file format serves as an application middleware. This library is responsible for accessing the common H5 file format. Additionally, when using G-FAST in a real-time scenario, this library has the necessary functionality for reading the Earthworm rings and and updating the HDF5 data file. Note, the CMake has been designed so that the Earthworm can quickly be ejected if it one only desires to use G-FAST in a playback mode or implement another datasource such as SeedLink. This library is summarized at a high level in a real-time application by Figure 2.

3.6 Moment Tensor Decompositions with cMoPaD

At the time of writing G-FAST is the only proposed ShakeAlert module which can perform moment tensor inversions. A common feature in CMT inversions is the ability to decompose the inverted moment tensor into isotropic, compensated linear vector dipole, and double couple parts (e.g. **?**). Towards this end a flexible Python package was introduced by **?** in MoPaD. ISTI has reimplemented the very general decomposition component of MoPaD in C for computational efficiency. The primary utility of such a decomposition library in G-FAST is in the CMT gridsearch double-couple percentage weighting. This library is also used in generation of a more complete QuakeML which describes a moment tensor in terms of fault planes as well as it's eigenvectors.

One should observe ISTI's cMoPaD is by definition derivative work. Consequently, this library must be distributed under the L-GPL license. The consequence is that any software statically linking to this library must have a license which is L-GPL compatible.

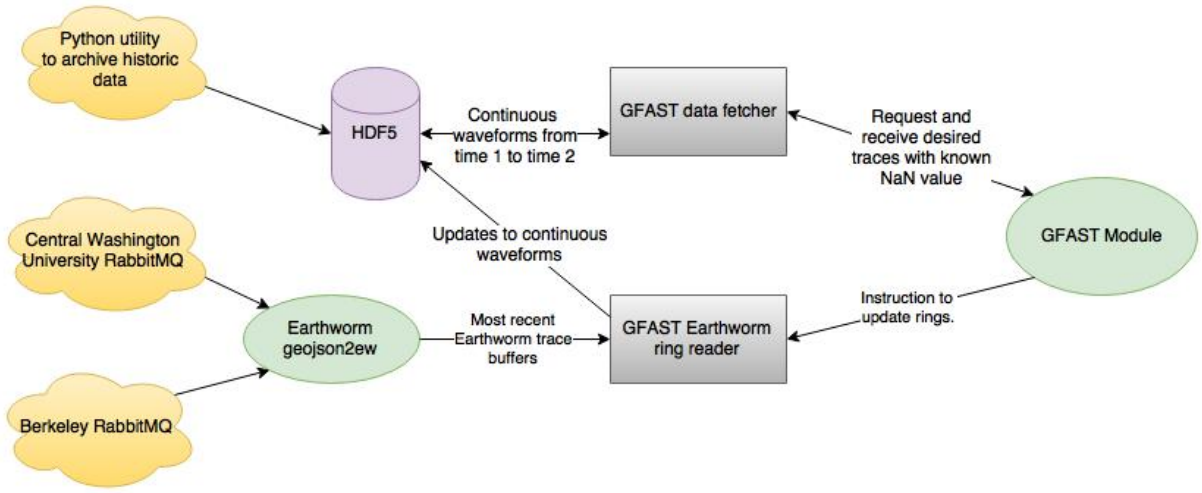

Figure 2: A depiction of the current data flow. The goal is to put the data in a common, continuous, finite-sized format hereby encapsulated by the HDF5 archive. In the context of playbacks the bottom functions are not used. Hence, it is the same function that queries the HDF5 archive in playback and real-time mode and the G-FAST application developer is only exposed to one data fetcher API. The API is very simple in that G-FAST specifies two times, for example the event origin time and the current time, and the data is returned with the understanding some data may be latent or non-existent and have a predetermined value indicating its absence. For real-time scenarios the Earthworm acquisition must be configured and running prior to launching a G-FAST application. The GPS data ring will be accessed directly by a G-FAST function that relies on libew.a. Real-time applications must additionally call the G-FAST ring reader function periodically, say every second, to refresh the HDF5 archive.

3.7 Data Acquisition

This section outlines the data acquisition strategy in real-time and playback operations. To make the real-time acquisition and playback appear integrated to the user a common HDF5 file format is selected. For playbacks, a pre-processing program can be used to generate a valid input data file[3] which is queried by a simple API for data in a given a time range. Likewise, real-time data is written[4] to an HDF5 file with the same structure as a playback data file and, like in playback mode, can be queried by the same API as the playback for data in a given time range.

[3]'File' is a generic word in the context of this discussion. It is unnecessary and potentially undesirable from a performance standpoint that the file physically exist in the computer's disk space. The illusion of keeping the file on disk is accomplished by instructing HDF5 to memory map the data file during the start-up phase.

[4]The writes can be to memory or disk.

3.8 XML Output

This section outlines the three XML data products. Two XML products, PGD and finite fault, have defined ShakeAlert schemas. The third XML product is a QuakeML summary CMT summary for potentially expediting communication between the tsunami warning centers and NEIC.

3.9 HDF5 Output

This section outlines the HDF5 summary files. HDF5 is a portable and high performance utility which makes it possible to easily archive the results[5] of G-FAST after each iteration thereby providing a snapshot of the GFAST execution and

4 Using G-FAST

This section describes the G-FAST usage in layback application and earthquake early warning. There are three steps in a G-FAST. The first step is creation of a parameter file and additional data files. The second step is running the application. The third step is post-processing the resulting HDF5 summary file.

4.1 The G-FAST Base Properties File

This section describes the G-FAST parameters common to both playback and earthquake early warning.

The most recent metadata file can be obtained at `http://www.panga.cwu.edu/eew/readi_sites_metadata.txt`.

4.2 The G-FAST Playback Properties File

4.3 The G-FAST Earthquake Early Warning Properties File

5 Acknowledgements

This software was developed with funds from the amazon catylyst and government grants a, b, c and d.

A License

brendan, you have to email the comotion (tech transfer) and see what they want. i'm going to guess this software belongs to uw, ideally it is free (as in beer) for academics and government organizations, you can freely modify the code however uw owns all modifications to the software, you cannot redistribute the code, the software is provided as is, and you cannot hold uw liable. i don't think gpl will be a good license choice. if this library is to exist in the shakealert framework gpl licensing can very easily find itself conflicting with caltech licensed software. but that's the usgs's problem and not yours.

[5]Since results are encapsulated in data structures the HDF5 files are the data structures after each iteration.