Rawspec

Testing Plan

Richard Elkins

2021-11-10 (v4)

https://github.com/UCBerkeleySETI/rawspec_testing

Rawspec Current State

- First stage in multiple production work flows that generate science products for downstream processing and analysis.
- Rawspec github repository: https://github.com/UCBerkeleySETI/rawspec
- Pull Requests
  - Proposed bug fixes and feature enhancements.
  - Unknown side effects (if any) on the current source base.

Testing Requirements

- Input data coverage:
  - Single polarization GBT .raw files.
  - Dynamically generate multipolarisation data (rawspectest)
  - ATA-supplied specialised files for multi-antenna, incoherent summing, and 4bit data.
- Assistance from Matt Lebofsky for the GBT files.
- Automated repeatable testing scripts:
  - Generate the Baseline Data, a collection of correct results, based on known stable versions of rawspec, blimpy, and turbo_seti.
  - PR Tryout preparation (git clone PR, make, etc.).
  - Generation of Trial Data using the PR code.
  - Evaluate the Trial Data against the Baseline Data.

Testing High-level Design

Prior to Entertaining Pull Requests, we need correct results in the Baseline Data.

<u>Generate the Baseline Data</u>

– Using the turbo_seti top hit table (.dat), generate a table of selected fields (frequency, drift rate, etc.).

  0000.raw → .fil → .h5 → .dat → .tbl_dat

  0000.raw →→→ .h5 → .dat → .tbl_dat

– Using the Filterbank header information, generate a table of (key, value) pairs (fch1, foff, etc.).

  .h5 → .tbl_hdr

– Discard intermediate files (.fil, .h5, .dat, and .log).

– Run rawspectest for 8- and 16-bit data.

  - Redirect stderr and stdout to files.
  - Make sure that stderr is empty.
  - From stdout, convert the “output product” text lines into .tbl_npols entries.

Testing High-level Design (cont.)

PR Tryout

<u>Generate the Trial Data</u>

Follow the Baseline Data generation steps but target the Trial Data.

<u>Evaluate the Trial Data against the Baseline Data</u>

- Compare <u>trial</u> .tbl_hdr files to their counterparts in the <u>baseline</u>.
- Compare <u>trial</u> .tbl_data files to their counterparts in the <u>baseline</u>.
- Compare the <u>trial</u> .tbl_npols file to its counterpart in the <u>baseline</u>.
- Report success and failure instances.

Note about comparing: Use *numpy.isclose(…, rtol=...)* for floating-point elements.

Testing Detailed-level Design Prerequisites

- Login to any data centre compute node.
- If the rawspec_test repository has not yet been installed,

  go to $HOME and `git clone …..`.
- Make sure that up-to-date blimpy and turbo_seti are installed under $HOME:

  `pip install -U --user blimpy`

  `pip install -U --user turbo_seti`

Note that none of this is dependent on a particular PR.

Testing Detailed-level Design (cont.)
Generating the Baseline Data

IMPORTANT: This procedure is unnecessary for PR testing and has the potential to be disruptive.

- Login to any data centre compute node.
- Go to $HOME/rawspec_testing/exec
- Run bash xinstall.sh

Testing Detailed-level Design (cont.)
New Pull Request

- Evaluate the PR features, potential impact, and how it fits in the existing software base.
- Login to any data centre compute node.
- Go to $HOME/rawspec_testing/exec
- Edit the xprep.sh script to supply:
  - PR URL value for its default branch which should end in "…../rawspec".
  - Specific BRANCH name value.
- Run bash xprep.sh. In the end, rawspec should have been built in $HOME/rawspec.
- Run bash xtest.sh *<GPU_ID (0, 2, 3, or 3)>*. Hopefully, you will see the following message at the end:

```
hh:mm:ss  reviewer  INFO  *SUCCESS* - No errors reported.
```

Note that the bash scripts take care of the following:

```
export PATH=$HOME/rawspec:$PATH
export LD_LIBRARY_PATH=$HOME/rawspec
```

Next Steps

- One last PR (#22) to be reviewed. Then, merge it as it has already been regression tested.
- Fork rawspec for integrating the FBH5 amendments.
- For all future PRs, follow the testing plan.