ver0: SPOTLIGHT's Ver0 Pipeline

ver0 is the zeroth version of **SPOTLIGHT**'s transient search pipeline. This pipeline is a temporary measure, until a fully online real-time pipeline is put into place. It is entirely offline, and is multi-beam, multi-node, and multi-GPU. It is only meant to run on the **Param Brahmand** system at the GMRT. Currently, it is being used to process data being collected as part of SPOTLIGHT's commensal operations in **GMRT's GTAC Cycle 48**.

Quick Start

To run it, follow these steps:

- Login in to the **Param Brahmand** system as the `spotlight` user.
- Navigate to the `tdsoft` directory (`cd /lustre_archive/apps/tdsoft`).
- Source the `env.sh` file (`source env.sh`).
- Enter the `ver0/` directory (`cd ver0` *or* `cd $VER0DIR`).
- Add the observations you wish to process to `assets/gtac.list`.
- Then run: `./ver0 run -a` OR `./ver0 run -r` OR `./ver0 run -m`. (See @notes.)

A few notes:

1. Before you run `ver0`, you need to fill in `assets/gtac.list` with the name of the observations you wish to process. These are the directory names for each observation, as recorded at `/lustre_data/spotlight/data`, with one directory name on each line. The pipeline will go through them one-by-one, and process all scans in each observation sequentially until it is done.

2. Note that two pre-existing node lists are already provided for you:

   - `assets/nodes.list.all`: A list of all 36 nodes available on Rack 1 and Rack 2. This list is only supposed to be used when the correlator is off, which is when all these nodes are available for offline processing.
   - `assets/nodes.list.ltd`: A list of 4 nodes in Rack 2: `rggpu36`, `rggpu37`, `rggpu38`, and `rggpu39`. These are the nodes which are available regardless of whether the correlator is operational or not. This allows us to run the pipeline alongside the correlator.

   `ver0` has the ability to pick one of these subsets. By default, it will pick all 36 nodes to run the pipeline; one can explicitly select this mode by running `./ver0 run -a` or `./ver0 run --all`. This checks if the correlator is running, and will prevent it from running if it is not already doing so. If one wishes to run the analysis on just the 4 nodes in `nodes.list.ltd`, then one should run the pipeline with `./ver0 run -r` or `./ver0 run --restricted`. Note that this will run whether or not the

correlator is running or not, since the nodes it uses are not used by the correlator. In case a different set of nodes is required to run the pipeline, one should manually edit the `assets/nodes.list` file before running the pipeline, and then run it with `./ver0 run -m` or `./ver0 run --manual`.

3. Since the pipeline does take some time to run, one can run it inside a detached terminal via `screen`. To do so, create a new terminal using `screen -S <NAME>`, source the `env.sh` file, run `ver0.sh` as instructed above, and then detach by pressing `Ctrl + A` and then `D`. To reattach, just run `screen -r <NAME>`, where `NAME` is the name you gave to the terminal when you created it. This can prevent a disturbance in, or disconnection of, the SSH connection from killing or interrupting the pipeline's run.

Usage

For more help on the `ver0` script, just run `./ver0` *or* `./ver0 help`. It should show something like this the screenshot displayed in Figure 1.

From Figure 1, we can see that the `ver0` script provides the following commands:

- **`run`**: Runs the pipeline. Requires no input. The user should fill the `gtac.list` file with the GTAC observations they wish to process, and then run this command as : `./ver0 run`. The pipeline will automatically pick up multiple scans from each observation (if present), and process them sequentially. It will then proceed to analyse the next observation. No intervention is required from the side of the user.
- **`kill`**: Kills the pipeline. The pipeline uses background processes run via SSH, and so the pipeline does not die immediately if interrupted via `kill`, or via `Ctrl + C` (that is, `SIGINT`). Thus, to *actually* kill the pipeline, one must run `./ver0 kill`.

> ⚠ Warning
>
> Note that the `ver0 kill` command kills all processes run by the `spotlight` user in each compute node listed in `nodes.list`. **THIS SHOULD BE USED CAUTIOUSLY**, lest it kill the SPOTLIGHT correlator as well.

- **`verify`**: Verifies that a run of the pipeline has gone through as it should. This checks if:
  - The filterbank files have been extracted properly from the raw files.
  - All filterbank files have been analysed successfully by `AstroAccelerate`.
  - Features have been extracted for all candidates from each beam.
  - All candidates from all beams have been successfully classified.

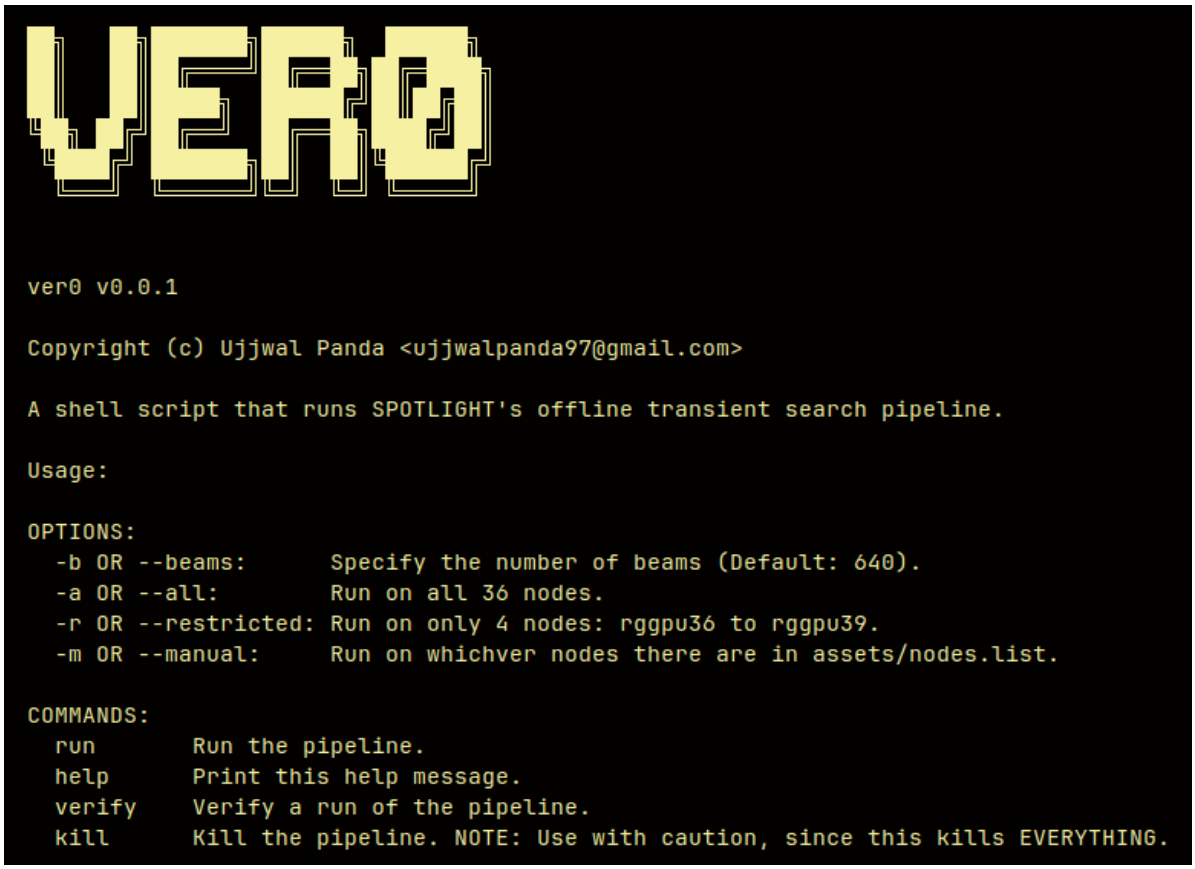

Figure 1: A screenshot of the help message for `ver0`, displayed when you just run it without any arguments, or if you run it as `./ver0 help`.

In case any of these processes fails, the user is informed.

- **help**: Show the help message shown above in the screenshot.

Details

Multiple beams are dumped from a ring buffer to disk, time sliced and concatenated together into a single raw file. For each observation, a directory is create according to its GTAC code, date, and time, in `/lustre_data/spotlight/data`. The raw files are dumped into the `BeamData` subdirectory. The `xtract2fil` package then extracts each beams and dumps it into a separate filterbank file in the `FilData` subdirectory. The pipeline then distributes the jobs across the nodes specified in `nodes.list`, and their individual GPUs, using the `distribute.py` script. This information is stored in text files in `/tmp`, for both the *pre* (essentially all of **AstroAccelerate**) and *post* (clustering + feature extraction + classification) stages. Then, both the *pre* and *post* stages are run. Dedispersion, single pulse search, peak filtering, and so on are carried out using the **AstroAccelerate** pipeline. Candidates are then clustered in `cluster.py`, features are extracted from each of them using **candies** in `candify.py`, and then each candidate is classified using **FETCH** in `classify.py`. The pipeline outputs and logs are dumped into the `FRBPipeData` subdirectory. In case the observation consists of multiple scans, the `FilData` and `FRBPipeData` subdirectories will have directories for each individual scan, and each of these directories will have the filterbank files and pipeline outputs respectively for each scan.

Directory Structure

A typical directory structure looks like this:

```
48_116_20250418_021733
    BeamData/
    FilData/
      SN2004DK_20250418_041550/
        BM0/BM0.fil
        BM1/BM1.fil

        BM639/BM639.fil
    FRBPipeData/
      SN2004DK_20250418_041550/
        BM0/
          curfile.txt
          candidates.csv
          global_peaks.dat
```

```
  peak_analysed-t_0.00-dm_0.00-153.60.dat

  peak_analysed-t_1313.57-dm_1536.00-2048.00.dat
  filtered_candidates.csv
  MJD60782.9482027_T1027.4524000_DM55.90000_SNR49423.81200.h5

  MJD60782.9482027_T999.2916000_DM0.60000_SNR10.35349.h5
BM1/...

BM639/...
classification.csv
VER0.rggpu36.0.txt
VER0.rggpu37.0.txt
VER0.rggpu38.0.txt
VER0.rggpu39.0.txt
VER0.rggpu36.1.txt
VER0.rggpu37.1.txt
VER0.rggpu38.1.txt
VER0.rggpu39.1.txt
```

The following files are dropped as part of the pipeline's output:

- **A `global_peaks.dat` file**: Contains all the candidates from the single pulse search carried out via **AstroAccelerate**. This is a binary file, consisting of a list of 32-bit floating point numbers, with 4 values (the dispersion measure or DM, the arrival time, the SNR, and the width) per candidate.
- **A `candidates.csv` file**: Same as `global_peaks.dat`, just converted to a CSV file.
- **A `filtered_candidates.csv` file**: Contains the list of candidates after clustering.
- **A number of `*.h5` files**: Features for each candidates.
- **A `classification.csv` file**: Output from classification.

Note that the `classification.csv` file combines the output from all beams, and is placed outside the beam subdirectories.