ver0: SPOTLIGHT's Ver0 Pipeline

This 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 currently 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. To run it, follow these steps:

- Login in to the **Param Brahmand** system as the `spotlight` user.
- Navigate to `/lustre_archive/apps/tdsoft`.
- Source the `env.sh` file.
- Enter the `ver0/` directory.
- Then run: `ver0.sh <GTAC_CODE>`.

Note that `ver0` expects the following directories to be present:

- `VER0_DIR`: The directory where the code for `ver0` itself is present.
- `VER0_BEAMS`: The directory where the raw data is expected to be present.
- `VER0_DATA`: The directory where the filterbank files are kept.
- `VER0_DISTS`: The directory where workload distributions are stored.
- `VER0_LOGS`: The directory where the logs are dumped.
- `VER0_OUTPUT`: The directory where the pipeline's output is dumped.

Multiple beams are dumped from a ring buffer to disk, time sliced and concatenated together into a single raw file. The `xtract2fil.py` script then extracts each beams and dumps it into a separate filterbank file in the `$VER0_DATA` directory. The pipeline then distributes the jobs across the nodes specified in `nodes.list`, and their individual GPUs, using the `distribute.py` script. This information is stored in text files in the `$VER0_DISTS` directory 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. We then cluster candidates (via `cluster.py`), extract features from them using **candies** (via `candify.py`), and classify them using **FETCH** (via `classify.py`). The pipeline logs are dumped to `$VER0_LOGS`, and the outputs to `$VER0_OUTPUT`. As indicated by the `$` sign, the path to each of these directories is specified as an environment variable in `env.sh`, which has to be sourced before the pipeline is run.

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.

A few more notes:

1. The pipeline expects the data to be present in a sub-directory in `$VERO_BEAMS`, which should be named after the **GTAC** (**G**MRT **T**ime **A**llocation **C**ommittee) code under which the observation was taken. This is also given as input to the pipeline when running it, and is exported as an environment variable, `$VERO_GTACCODE`, during each run. The outputs are similarly dropped inside `$VERO_OUTPUT/$VERO_GTACCODE/$VERO_JOBID`, wherein the last sub-directory is also exported as an environment variable, and is actually the timestamp when the pipeline is run.
2. The pipeline does not plot the final output(s) for now. One can plot the `*.h5` files using `candies`. Just run `candies plot *.h5`. For the other outputs, a simple knowledge of Python should suffice :D.
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.