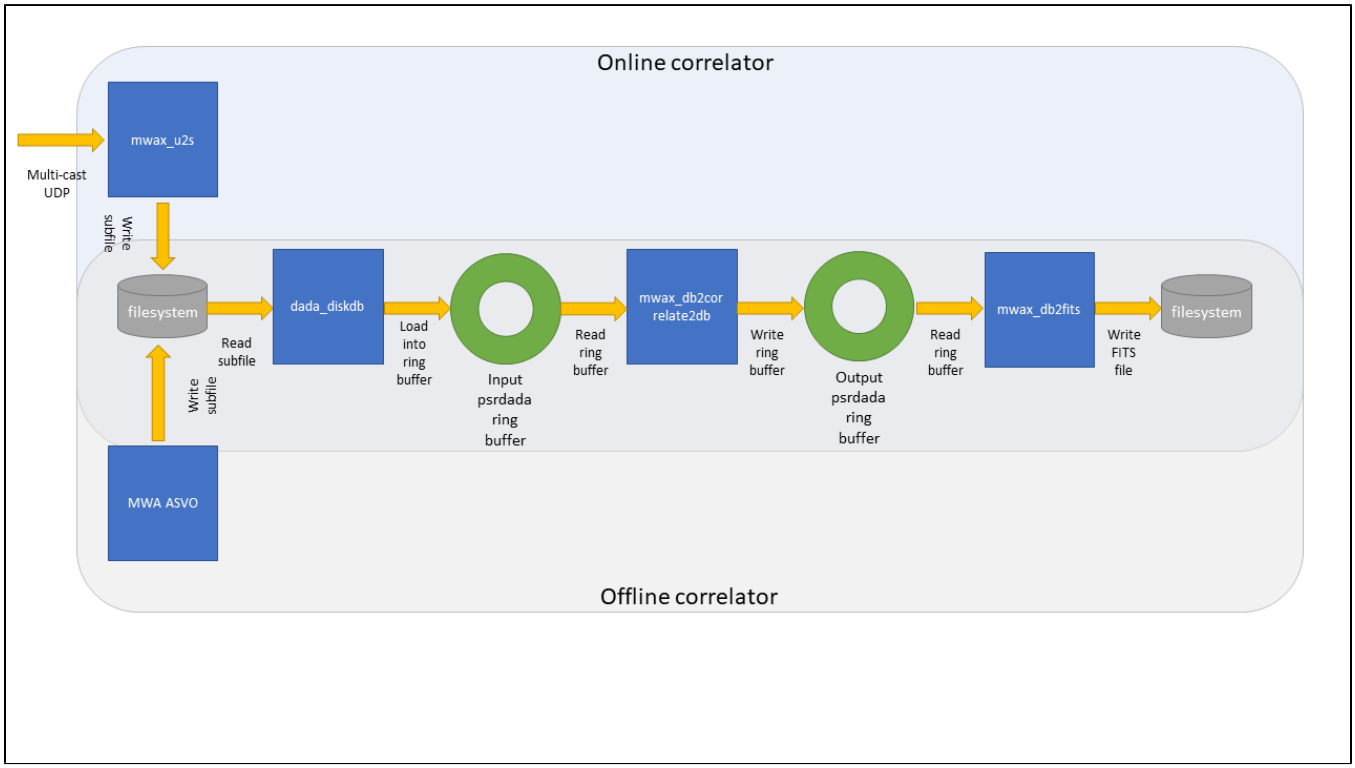
MWAX Offline Correlator

Overview

The MWAX correlator was built to be modular and to be run real-time in production at the MRO, or offline using archived voltage subfiles instead of live multicast UDP packets.



Hardware

CUDA GPU

A significant fraction of the xGPU library and the mwax_db2correlate2db process is written in CUDA, so a CUDA-compatible GPU is essential.

Software

CUDA

Visit the Homepage. CUDA is required to be installed prior to PSRDADA, and mwax_xGPU.

CFITSIO library

Visit the Homepage.

```
$ CFLAGS="-O3" ./configure --prefix=/usr/local --enable-reentrant --enable-sse2 --enable-ssse3 --disable-curl
$ make clean && make
$ make install
```

PSRDADA library

Visit the Homepage

The MWAX correlator uses, and has tested, this commit (ca505cdb519afbd63ae91c00e4d86af0f3313b68) from the PSRDADA repository.

For Ubuntu 20.04 systems, the following packages also need to be installed in order to compile PSRDADA successfully (this was discovered through trial and error) - other Debian based systems should have similar requirements:

```
$ sudo apt update
$ sudo apt install automake autoconf csh hwloc libhwloc-dev libhwloc-plugins libtool numactl pkg-config python2.
7-dev python
```

This is how we build it (note: some paths may be different on your system):

```
$ ./bootstrap
$ ./configure --with-cuda-dir=/usr/local/cuda --with-cuda-lib-dir=/usr/local/cuda/lib64 --with-hwloc-lib-dir=
/usr/lib/x86_64-linux-gnu --prefix=/usr/local
$ make
$ make install
```

If all goes well, you should be able to run commands, such as:

```
$ dada_db --help
```

mwax_xGPU library

Visit the github page: https://github.com/MWATelescope/mwax-xGPU

The mwax-xGPU fork must be compiled with very specific flags in order to work. This is for a 128T system:

```
$ make clean NSTATION=128
$ make CUDA_ARCH={{gpu_sm_arch}} NSTATION=128 NFREQUENCY=6400 NTIME=52 NTIME_PIPE=52
$ make install prefix=/usr/local NSTATION=128 NFREQUENCY=6400
```

Replace {{gpu_sm_arch}} with the correct SM_ARCH value for your GPU. See CUDA GPU List (wikipedia). Read off the ***"Compute capability (version)"*** value, remove the dot and prefix with "sm". E.g. an Nvidia V100 would be **sm_70**.

For cases where tiles >192T and up to 256T:

```
$ make clean NSTATION=256
$ make CUDA_ARCH={{gpu_sm_arch}} NSTATION=256 NFREQUENCY=6400 NTIME=56 NTIME_PIPE=28
$ make install prefix=/usr/local NSTATION=256 NFREQUENCY=6400
```

xGPU's NSTATION has a minimum of 16 and **MUST** be in increments of 16. So, for example, if you need to correlate 136T, then NSTATION needs to be 144 because 136 is not a multiple of 16. 144 is the next closest multiple.

mwax_common

Visit the github page: https://github.com/MWATelescope/mwax_common

This repo is required by mwax_db2correlate2db and mwax_db2fits. It contains some common header and c files which both programs require.

mwax_db2correlate2db

Visit the github page: https://github.com/MWATelescope/mwax_cbf

mwax_db2fits

Visit the github page: https://github.com/MWATelescope/mwax_db2fits

How to Run

Setup

Prepare input data

- Ensure all the subfiles you wish to process are in one directory e.g. /path/to/subfiles
- For each subfile you will need to modify the following Key/Value pairs to your use case:
- One way to modify these values is to use the mwax_update_subfile_header utility in the mwax_user_tools github repo.

Key	Existing Value	Set To	Notes
MODE	MWAX_VCS	MWAX_CORRELATOR	Required
INT_TIME_MSEC	a default set by the M&C system	250-8000	This is the correlator integration time (in ms)
FINE_CHAN_WIDTH_HZ	a default set by the M&C system	200-1280000	This is the correlator frequency resolution (in Hz). E.g. 10000 would be 10 kHz.
FSCRUNCH_FACTOR	a default set by the M&C system	1-6400	The number of 200 Hz ultrafine channels to scrunch together into a fine channel. E.g. for 10 kHz fine channels set this to 50
NFINE_CHAN	a default set by the M&C system	1-6400	Redundant info but must be compatible with FSCRUNCH_FACTOR and FINE_CHAN_WIDTH_HZ. E.g. 128 for 10kHz fine channels
EXPOSURE_SECS	the duration of the VCS observation	The duration of the correlator observation you want.	Must be in unit of 8 seconds, matching the total number of subfiles you wish to correlate per coarse channel * 8.
OBS_ID	The original obs_id of the VCS observation	The first SUBOBS_ID of the data you want to correlate	So if you have a VCS observation and want to correlate only from 80 seconds in, then you need to make OBS_ID==SUBOBS_ID==the gps time at 80 seconds into the VCS obs.
OBS_OFFSET	The 8 second offset for this subfile for the original VCS observation.	The offset you need for your correlator observation.	The first subfile in your correlator observation should be 0. Next will be 8, and so on.

See also: MWAX PSRDADA header for more information on the definition of each key within the subfile header.

Create Ring Buffers

```
# Create input ringbuffer
dada_db -b 32768000 -k 1234 -n 644 -l -p

# create output ringbuffer
dada_db -b 338297856 -k 2345 -n 16 -l -p
```

Launch Programs

```
#
# start mwax_db2fits- in offline mode you still need to provide health command line args even if you don't
intend to monitor the health packets
#
nohup mwax_db2fits/bin/mwax_db2fits -k 2345 --destination-path=/path/for/output/fits/files/. --health-
netiface=eth0 --health-ip=224.0.2.2 --health-port=8005 > mwax_db2fits.log 2>&1 &

#
# start mwax_db2correlate2db correlator
#
nohup mwax_cbf/mwax-fullcorrelator/mwax_db2correlate2db 1234 2345 224.0.2.2 8004 -a 5 -b 160 -d 0 -f 6400 -i
256 -o 1280 -O 2 -r -t -v -v > mwax_db2correlate2db.log 2>&1 &
```

Processing

> It is important to process each coarse channel separately. Do not mix coarse channel subfiles- process all of one channel before processing all of another channel. You should then have a FITS file per coarse channel

For this you will need a bash script to execute a **for/while loop** through all subfiles and for each one run:

```
dada_diskdb -k 1234 -f /path/to/subfiles/obsid_subobs_chNNN.sub
```

Each time this is executed it will fill the input ringbuffer with the subfile, and that will kickstart mwax_db2correlate2db processing data and filling the output ringbuffer with visibilities and weights, which will then kickstart mwax_db2fits writing out FITS files.

Clean up

```
# Clean up ringbuffers
dada_db -d -k 1234
dada_db -d -k 2345
```