NOEMA DiFX processing for EHTC

Issue and some thoughts up for discussion!

Jan Wagner, MPIfR Bonn

NOEMA

- Phased array of IRAM/France, 8-antenna 86-230 GHz
- Extension of the Plateau de Bure interferometer
- Bandwidth and hardware correlator upgrades

NOEMA (NOrthern Extended Millimeter Array)

NOEMA, the successor to the Plateau de Bure observatory, is the most powerful millimeter radiotelescope of the Northern Hemisphere and one of the most advanced facilities existing today for radio astronomy:
https://www.iram.fr/GENERAL/NOEMA-Phase-A.pdf

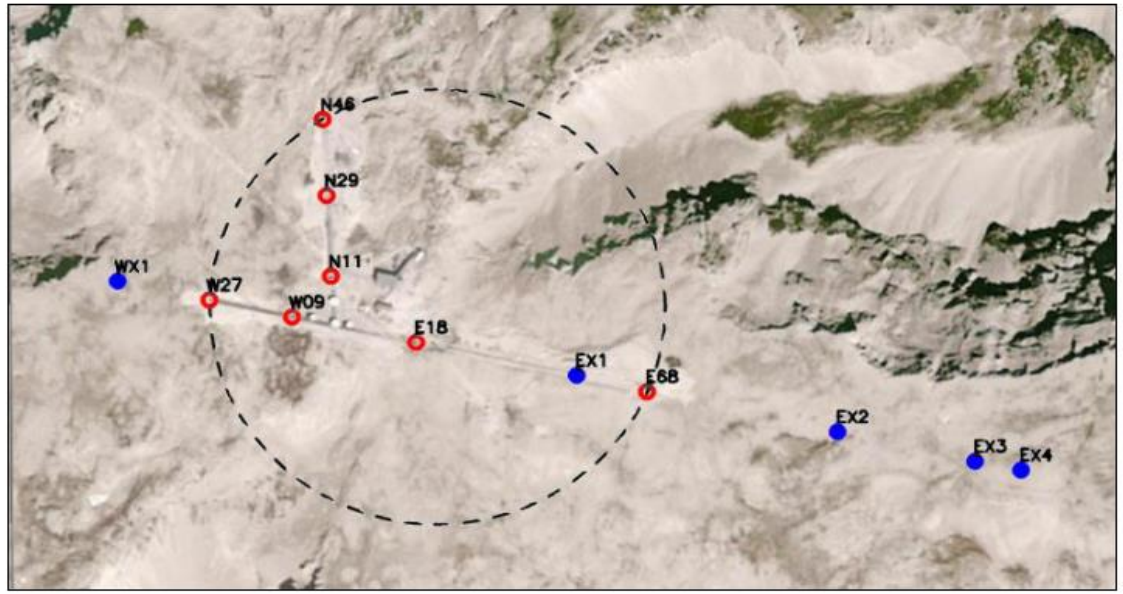
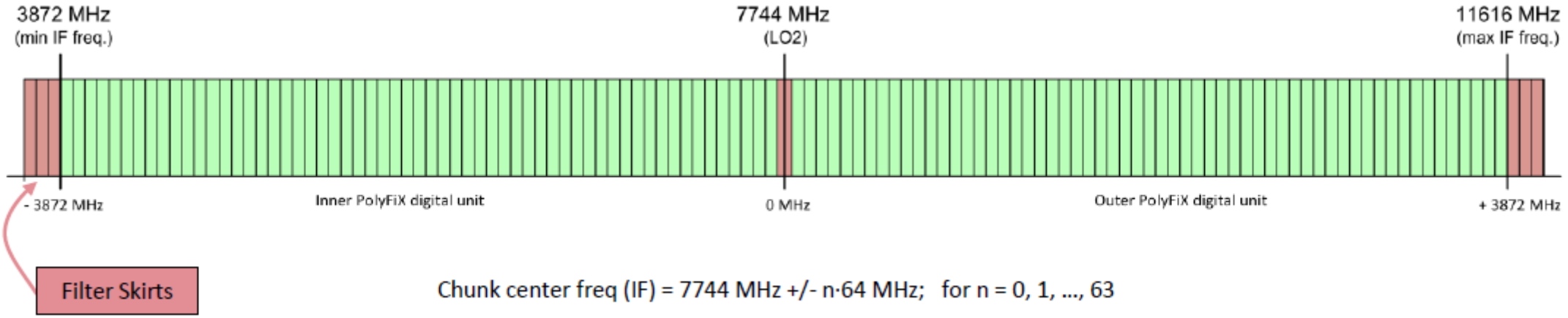


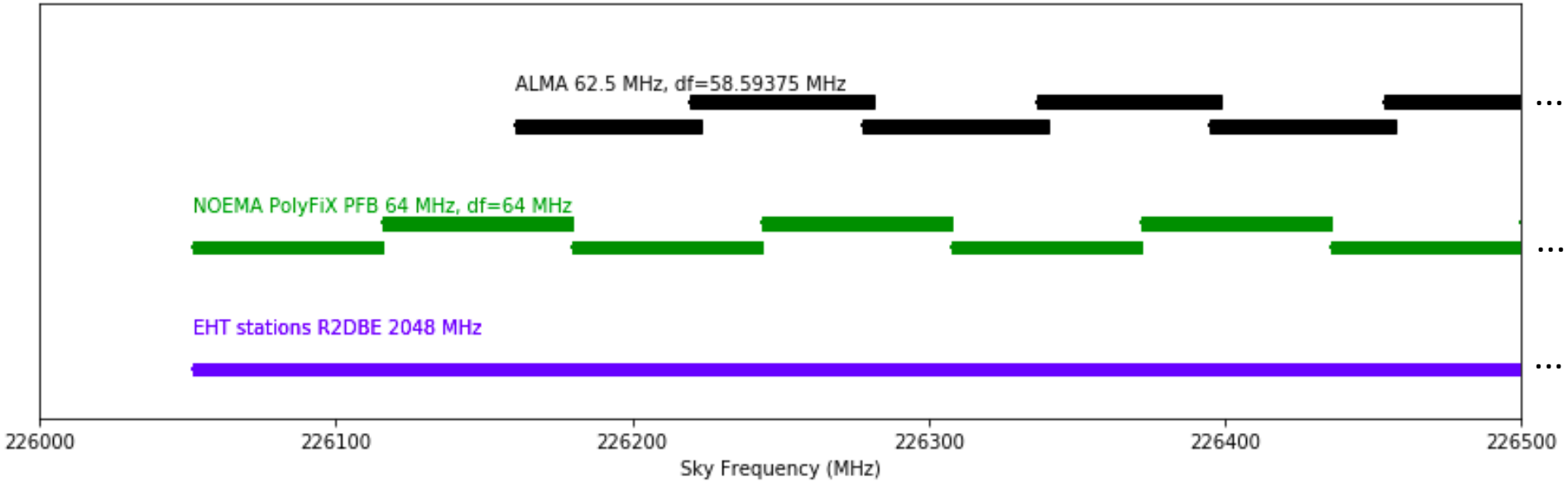
Fig. 11: The 12 locations of the A configuration pads (red: existing stations, blue: planned stations) overlayed on an aerial view of the Plateau de Bure Observatory (E is right, N is top). The tracks of the current PdB array are confined within a circle (dashed) of ~760 m diameter.

NOEMA Phasing Project

- R. Tilanus, H. Falcke, BlackHoleCam
- Hardware FFX-type PFB channelizer-correlator “PolyFiX”
- Correlator VLBI interface board to Mark6 recorders
- Recording H+V pol, USB+LSB, IF bandwidth 2-8 GHz, channelized



NOEMA to EHT/GMVA+ALMA: Bandwidth Mismatch



- 1 x 2048 MHz from EHT/GMVA wideband
- 64 ch x 64 MHz uniform channelization from NOEMA
- 32 ch x 62.5 MHz uniform *overlapped* channelization from ALMA
  - channel bandwidth: 62.5 MHz (20000 * 3125 Hz)
  - channel spacing: 58.59375 MHz (18750 * 3125 Hz)

Some options to make this “correlateable”

A. FPGA: modify NOEMA channelizer for wider channels e.g. 2048 MHz

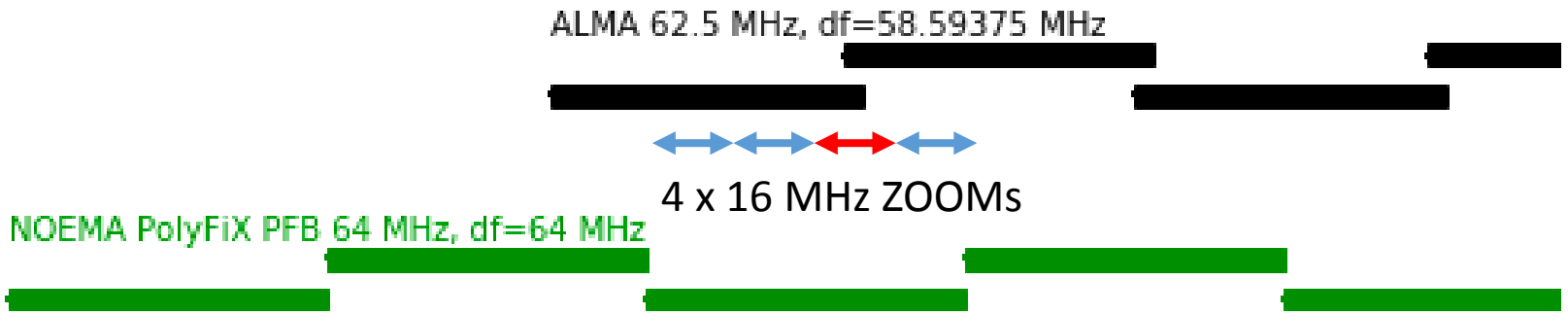
B. DiFX setup: narrow ZOOMs of say 16 MHz to partly cover 62.5/64 MHz

C. DiFX setup: use mixed-width ZOOMs and post-process in difx2difx.py

D. DiFX setup + modifications: mixed-width ZOOMs as in (C),
move difx2difx.py-like processing into *mpifxcorr*,
perhaps also Adams oversample/edge-crop -like work

Option B: DiFX with narrow ZOOMs

- Zooms of e.g. 16 MHz over EHT 2048 MHz x NOEMA 64 MHz x ALMA
- Maximally covers bandwidth of EHT x NOEMA PFB
- Partially covers bandwidth of ALMA x NOEMA (discards <30%)



- Results in large number of FITS/HOPS IFs (>64)

Option C: DiFX and mixed ZOOMs, difx2difx.py

- Cover NOEMA 64 MHz using overlapped ALMA 62.5 MHz channels, avoiding edges of ALMA channels i.e. avoid further bandpass effects

```
ZOOM NoemaWithAlma {
    ### additional zooms for covering NOEMA PFB with ALMA
    addZoomFreq = freq@226160.546875/bw@43.046875/noparent@true  # NOEMA ch02 x ALMA ch00
    addZoomFreq = freq@226219.140625/bw@20.953125/noparent@true  # NOEMA ch02 x ALMA ch01
    addZoomFreq = freq@226219.140625/bw@37.640625/noparent@true  # NOEMA ch03 x ALMA ch01
    addZoomFreq = freq@226277.734375/bw@26.359375/noparent@true  # NOEMA ch03 x ALMA ch02
```

- Correlation produces temporary .difx
- Temporary .difx has high spectra resolution of 15.625 kHz = GCD of ALMA x NOEMA zoom bandwidths, zoom vs band edge frequency offsets
- Difx2difx.py to derive final .difx(+.input) 64 MHz IFs at 0.5 MHz reso.

Overhead issues

- Some storage needed for temporary .difx before final .difx+.input
- Network load due to high-res visibilities from Core->FxManager
- Load in FxManager due to handling large volume high-res visibilities

- Slightly more complex processing: DiFX -> difx2difx -> PolConvert, although this could be made transparent and scripted

Crude benchmarking

- Correlated a 420s 9-antenna scan from EHTC 2017 in DiFX 2.5.2
- Used 58 MHz as target bandwidth (did not yet repeat with 64 MHz)

- Benchmark results

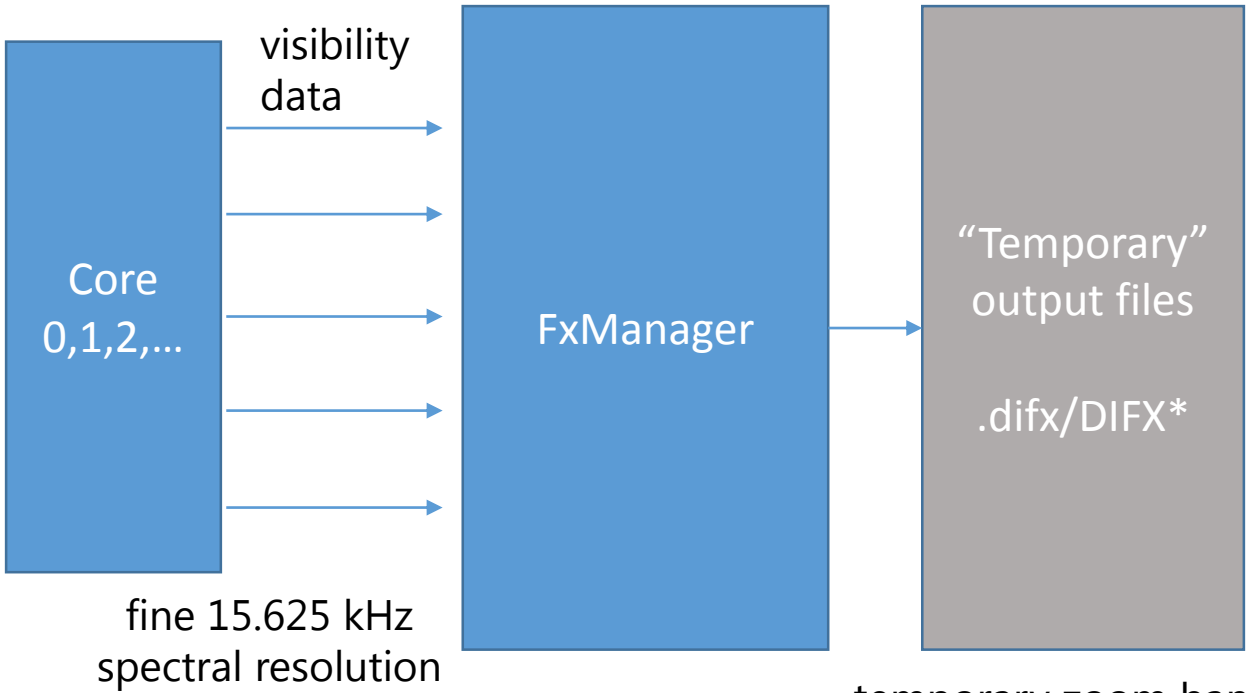
Correlation setup	exec time	size of temp .difx
1. 58 MHz zooms, 0.5 MHz out resolution	**~2000 sec**	**2 GB (reference)**
2. 58 MHz zooms, 15.625 kHz out	~4000 sec	68 GB
3. Mixed zooms incl. 58 MHz, 15.625 kHz	~6500 sec	131 GB
4. Mixed zooms w/o 58 MHz, 15.625 kHz	***~4000 sec***	***70 GB (opt. C)***

Option D: difx2difx-like processing in mpifxcorr

- Several possibilities to optimize and reduce overheads
- The associated changes might benefit other use cases?
  e.g. Adams oversample edge-crop, also general DiFX performance
- Some ideas (G. Crew, me):
  - No changes to DiFX. Run “converter” tools after correlation.
    -> better document and script the extra processing tools
  - Changes to DiFX of various degrees
    -> next slides…

Current situation

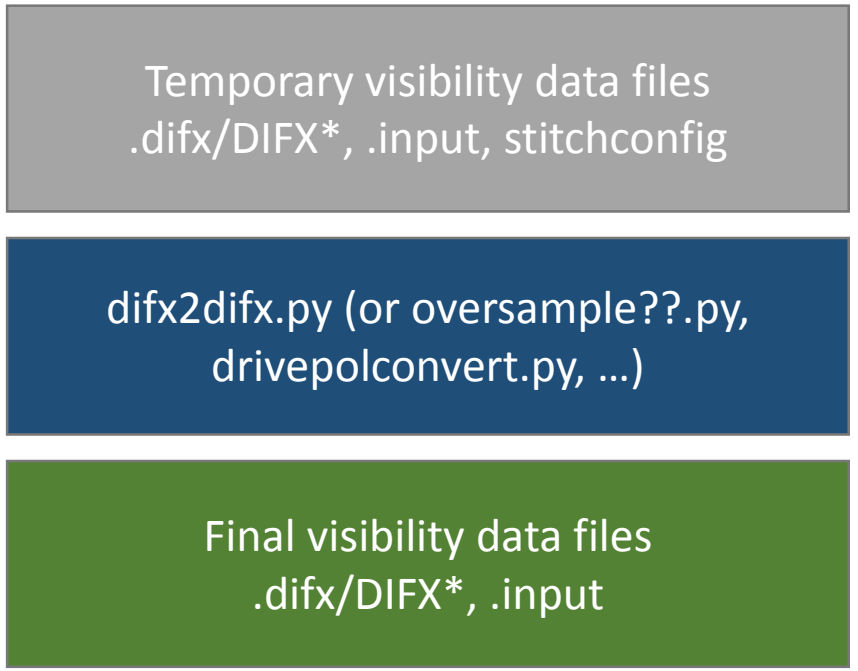
Step 1: Correlate with DiFX



fine 15.625 kHz
spectral resolution

temporary zoom bands
fine 15.625 kHz spectral resolution

Step 2: Post-process externally

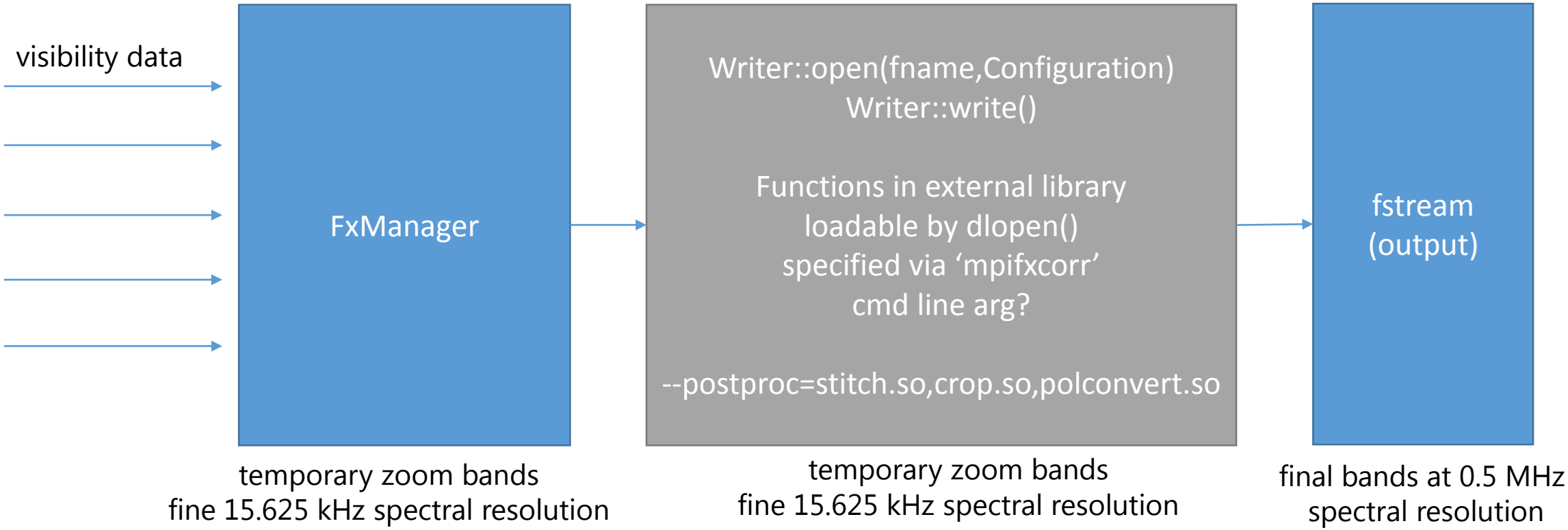


final bands at 0.5 MHz spectral resolution

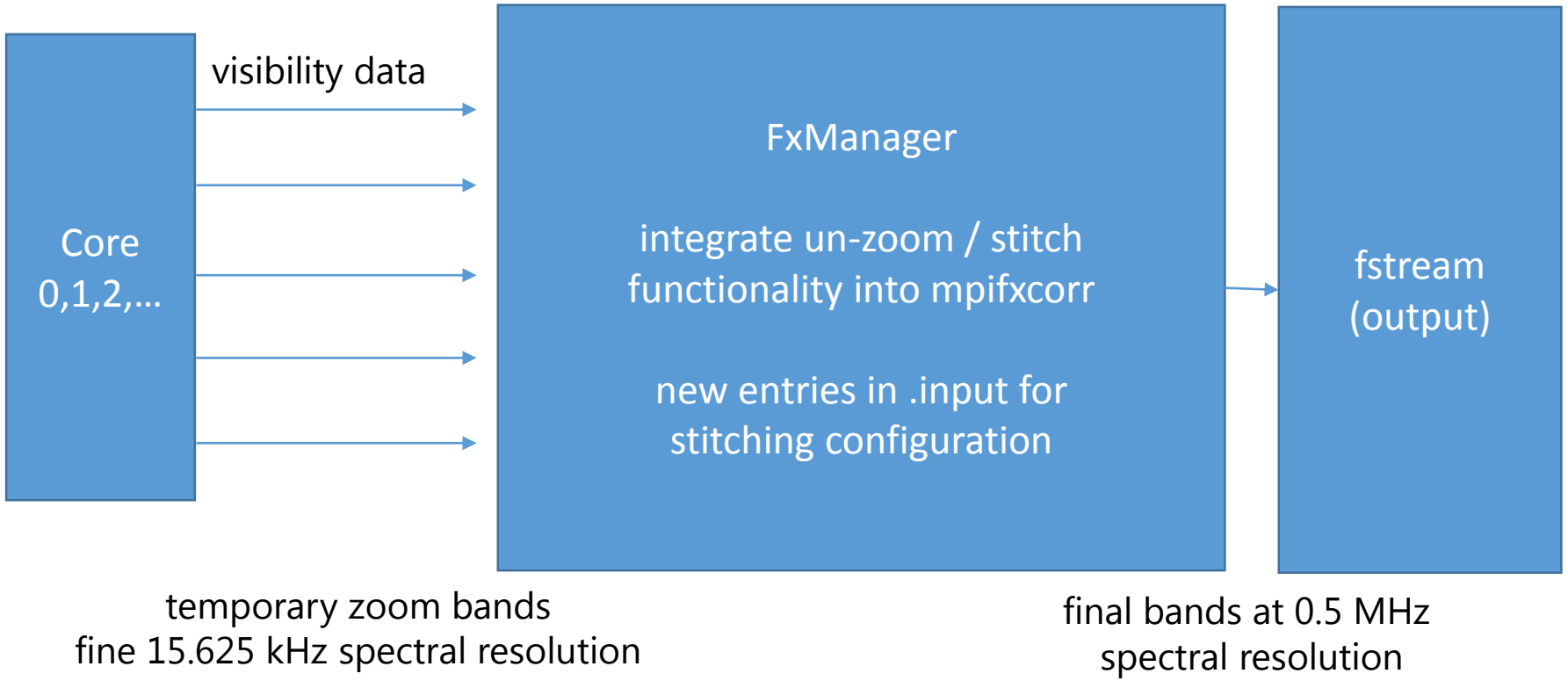
Post-processing moved into DiFX, functionality can be enabled on demand.

Could be a .so **library/plug-in** function call.

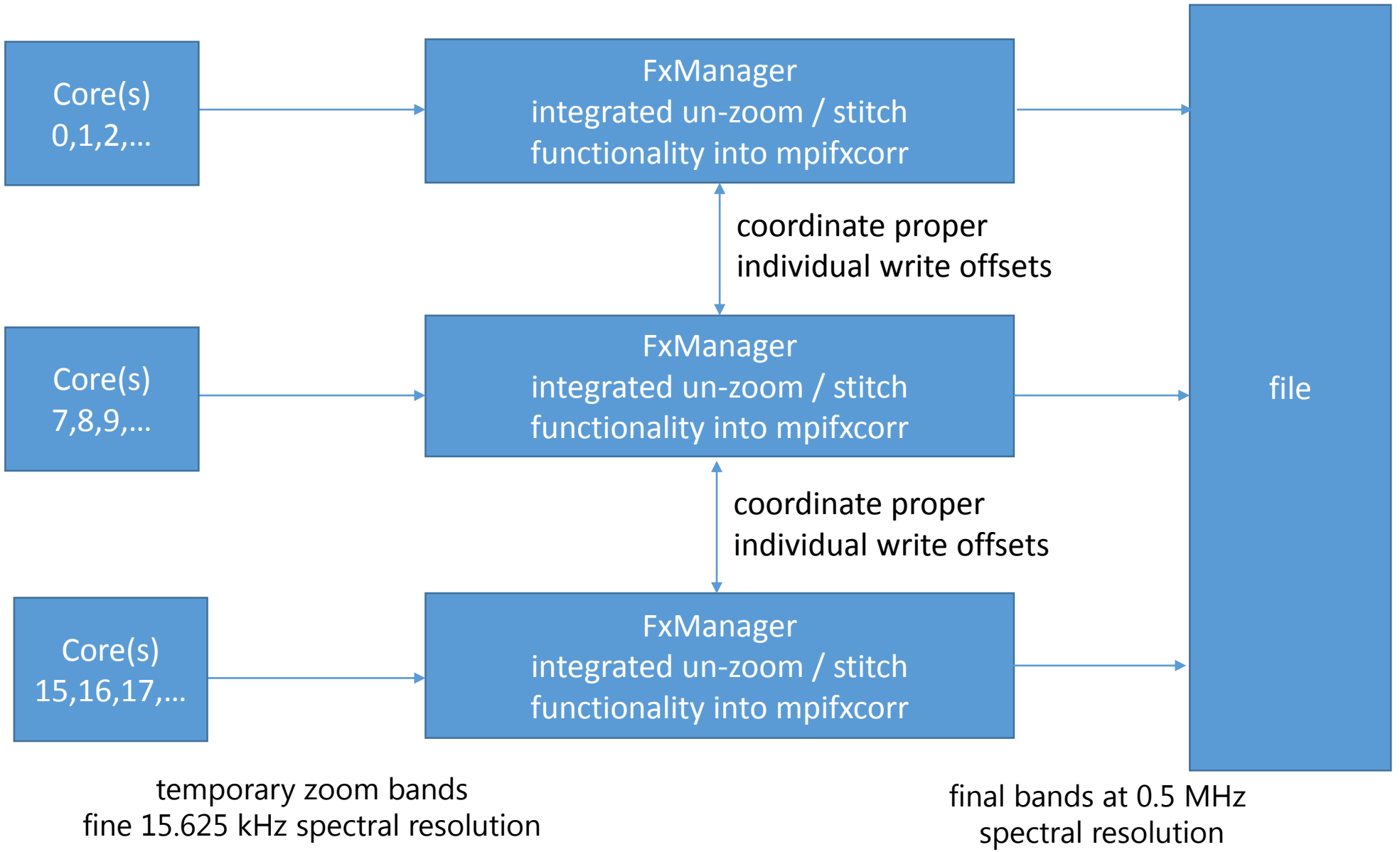
Could be a pipe to an **agent/program.**



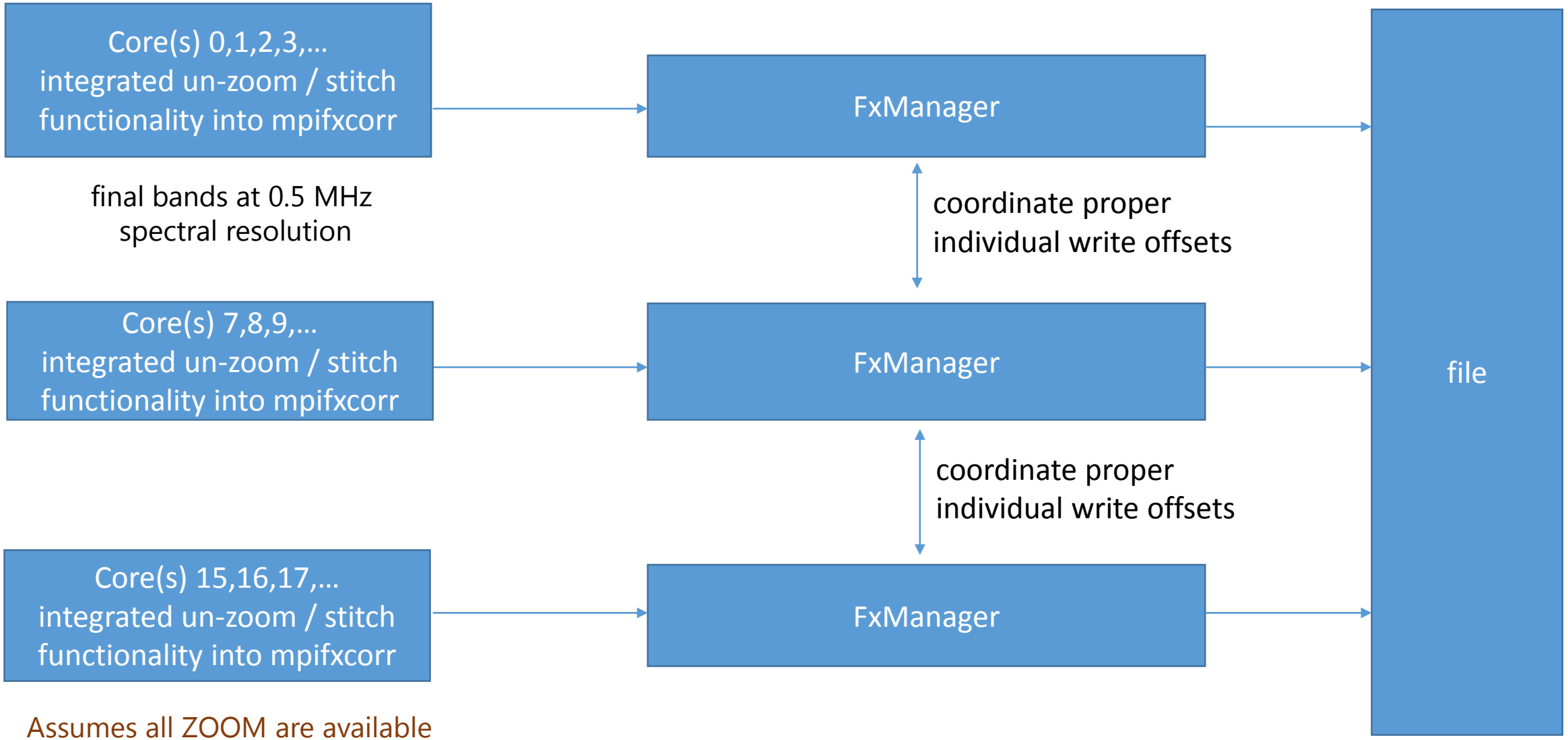
Functionality more tightly integrated (non-pluggable)



Reduced chance of Infiniband/IO bottleneck Core --> FxManager : distributed FxManager



Reduced chance of Infiniband bottleneck Core --> FxManager : spectral stitch in Core



Results of Discussion

Modifications

v2d

```
OUTPUTBANDS bandset1 {
   addOutputBand=lowfreqMHz/highfreqMHz
   addOutputBand=lowfreqMHz/highfreqMHz
}
DATASTREAM AntAA_DS1 {
   # not yet part of VEX, hence add to v2d
   oversamplingFactors=<num>/<denom> # e.g. 32/27
}
```

input

```
D/STREAM A BAND 0: 10=2,4,8
D/STREAM B BAND 0: 11=1,3,9
D/STREAM A CHANS 0: 16-2032,16-2032,16-2032
D/STREAM B CHANS 0: 16-2032,16-2032,16-2032
  with <freq>=<in_band0>,<in_band1>,<...>
  and bins <in_band0 startCh – stopCh>, <in_band1 startCh – stopCh>,..
```

Vex2difx v2d->input

1. parse new OUTPUTBANDS sections
2. replicate logic from Python to auto-introduce zoom bands where needed

Core/Mode

1. update to ‘threadscratchspace’ needed to hold combined output band data
2. new ::isFrequencyWritten() needed, what to send into procslots[idx].results[]
3. data weight of merge band has to be computed from N:1 contributing rec bands