Multi-Order HEALPix Map implementation in CDS HEALPix Rust

François-Xavier Pineau[1]

[1]Centre de Données astronomiques de Strasbourg

IVOA, College Park, 06 June 2025

Introduction

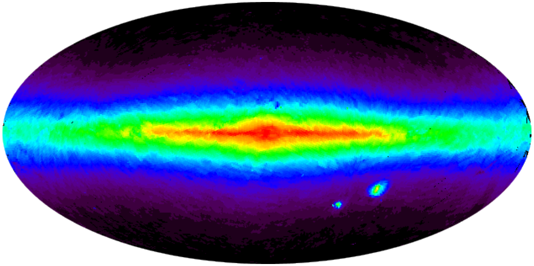
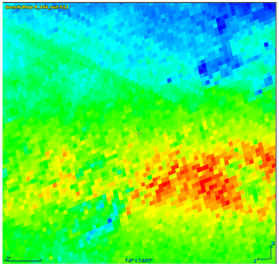
The HEALPix ecosystem encompasses various “maps” (often leading to confusion).

By chronological order (to my knowledge):

- **Skymap** (flat resolution HEALPix maps)
  - FITS convention from the official HEALPix lib
- **MOC**: **M**ulti-**O**rder **C**overage map
  - IVOA standard, v1.0 serialisation inspired from skymaps
- **BMOC**: **B**inary **M**ulti-**O**rder **C**overage map
  - limited to CDS HEALPix (Java and Rust)
- **MOM**: **M**ulti-**O**rder HEALpix **M**ap
  - Martinez-Castellanos, Singer et al. 2021, used in Ligo/Virgo
  - FITS serialisation also inspired from skymaps

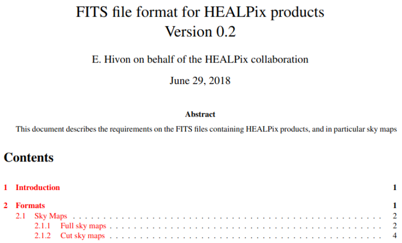
Skymaps (flat resolution)

- HEALPix maps, aka *Skymaps*
- Description:
  - list of **cells at a same resolution** (regular hpx grid)
  - a quantity/**value is associated to each cell**
- Examples:
  - Ligo/Virgo flat probability map
  - 2MASS count/density map

Skymaps (flat resolution)

- FITS serialisation (several conventions)
  - convention from the official HEALPix library
  - doc. asked by M. Taylor (TOPCAT), according to P. Fernique
    - v0.6 available here
  - `EXPLICIT` maps: `BINTABLE` with `IPIX` + values


FITS file format for HEALPix products
Version 0.2

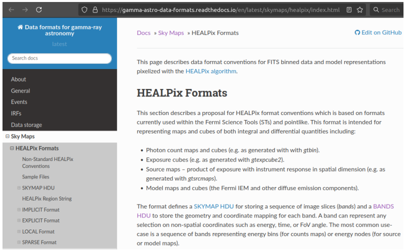
E. Hivon on behalf of the HEALPix collaboration

June 29, 2018

Abstract

This document describes the requirements on the FITS files containing HEALPix products, and in particular sky maps

Contents



This page describes data format conventions for FITS binned data and model representations pixelized with the HEALPix algorithm.

HEALPix Formats

This section describes a proposal for HEALPix format conventions which is based on formats currently used within the Fermi Science Tools (STs) and pointlike. This format is intended for representing maps and cubes of both integral and differential quantities including:

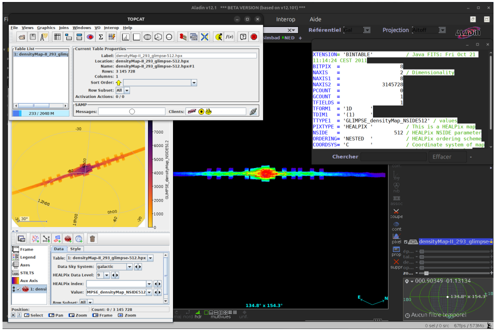
- Photon count maps and cubes (e.g. as generated with with *gtbin*).
- Exposure cubes (e.g. as generated with *gtexpcube2*).
- Source maps – product of exposure with instrument response in spatial dimension (e.g. as generated with *gtsrcmaps*).
- Model maps and cubes (the Fermi IEM and other diffuse emission components).

The format defines a SKYMAP HDU for storing a sequence of image slices (*bands*) and a BANDS HDU to store the geometry and coordinate mapping for each band. A band can represent any selection on non-spatial coordinates such as energy, time, or FoV angle. The most common use-case is a sequence of bands representing energy bins (for counts maps) or energy nodes (for source or model maps).

Similar but “non-offical” gamma-astro convention.

Skymaps (flat resolution)

- Well supported:
  - official HEALPix libraries (HEALPy), TOPCAT, Aladin, . . .
  - partly supported in CDS HEALPix/MOC Rust/Python (read in MOCPy; write `IMPLICIT` in CDS HEALPix Python)



Current Table Properties
Label: densityMap-II_293_glimpse-512.hpx
Location: densityMap-II_293_glimpse-512.hpx
Name: densityMap-II_293_glimpse-512.hpx#1
Rows: 3 145 728
Columns: 1
Row Subset: All
Activation Actions: 0 / 0

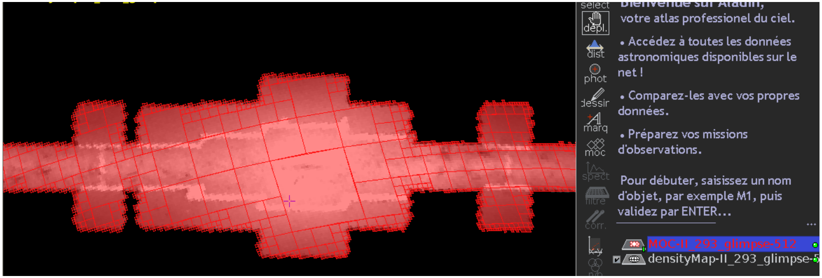
Table: 1: densityMap-II_293_glimpse-512.hpx
Data Sky System: galactic
HEALPix Data Level: 9
Value: MPSE_densityMap_NSIDE512
Count: 0 / 3 145 728

```
XTENSION= 'BINTABLE'           / Java FITS: Fri Oct 21
11:14:24 CEST 2011
BITPIX  =                    8
NAXIS   =                    2 / Dimensionality
NAXIS1  =                    8
NAXIS2  =              3145728
PCOUNT  =                    0
GCOUNT  =                    1
TFIELDS =                    1
TFORM1  = '1D      '
TDIM1   = '(1)     '
TTYPE1  = 'GLIMPSE_densityMap_NSIDE512' / values
PIXTYPE = 'HEALPIX '           / This is a HEALPix map
NSIDE   =                  512 / HEALPix NSIDE parameter
ORDERING= 'NESTED  '           / HEALPix ordering scheme
COORDSYS= 'C       '           / Coordinate system of map
```

Figure 1: Load Glimpse hpx file (T. Boch) in TOPCAT and Aladin

MOC

- MOC: **M**ulti **O**rder **C**overage map
- Description:
  - list of **non-overlapping cells at various resolutions**
  - 4 **siblings cells must be merged** into the parent cell
  - a MOC is a coverage/a mask on the unit-sphere
- Example
  - coverage of the glimpse survey



Bienvenue sur Aladin,
votre atlas professionel du ciel.

- Accédez à toutes les données astronomiques disponibles sur le net !
- Comparez-les avec vos propres données.
- Préparez vos missions d'observations.

Pour débuter, saisissez un nom d'objet, par exemple M1, puis validez par ENTER...

MOC

- MOC is an IVOA standard
- FITS serialization
  - v1.0 inspired from skymaps + uses `NUNIQ`
    - `NUNIQ`: `ORDER` + `IPIX` packed together
  - v2.0 serialisation relies on `ORDER` 29 ranges
    - faster set operations (not/and/or/xor)
    - no `NUNIQ` for Time and Frequency quantities
- For implementations/tools see IVOA MOC page
  - Java:
    - official/CDS library used in Aladin (P. Fernique, M. Reinecke), TOPCAT, . . .
    - official HEALPix lib (probably no v2.0 support)?
  - Python:
    - MOCPy (Rust based)
    - PyMOC (pure python), no v2.0 support (yet?)
  - Rust + WebAssembly: CDS MOC Rust (+moc-cli), MOCWasm
  - . . .

BMOC

- BMOC: **B**inary **M**ulti-**O**rder **C**overage map
- Description:
  - list of **non-overlapping cells at various resolutions**
  - a **binary value** (0 or 1) is **associated to each cell**
  - 0: the cell is partially covered; 1: the cell is fully covered
  - 4 siblings cells **having a same flag value** are merged into the parent cell
  - "special" set operations (not/and/or/xor)
    - flag 1 OR flag 0 = 1
    - flag 1 AND flag 0 = 0
    - NOT flag 0 = 0; NOT flag 1 = $\emptyset$
- Example
  - see Fast (possibly complex) STC-S queries thanks to B-MOCs

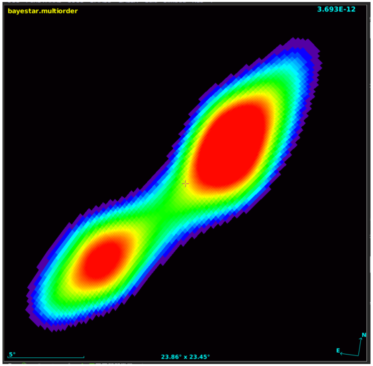
BMOC

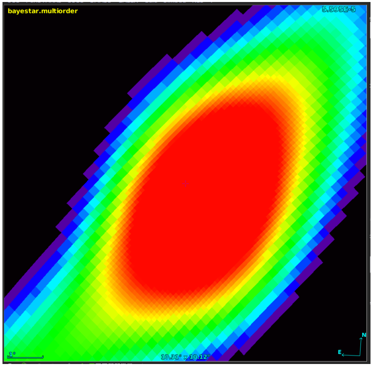
- No document, no paper
  - only implemented in CDS HEALPix Java / Rust
  - relies on `ZUNIQ`, same trick as in google s2geometry
  - **New!** command line tool hpx-cli
- **New!** Currently 2 FITS serializations
  - both uses `ZUNIQ` numbering, must be ordered
    - one custom: BIGENDIAN in primary HDU to optimize FITS size and allows for memmap
    - one interoperable: BINTABLE with `ZUNIQ` and flag columns
- BMOC from the command line with hpx-cli:

```
> hpx cov 9 stcs "Circle ICRS 147.6 69.9 0.4"
> hpx cov -t fits -o bmoc.fits \
  9 stcs "Circle ICRS 147.6 69.9 0.4"
> hpx cov -t bintable -o bmoc.fits \
  9 stcs "Circle ICRS 147.6 69.9 0.4"
```

MOM

- MOM: **M**ulti **O**rder sky**M**aps
- Description:
    - list of **non-overlapping cells at various resolutions**
    - a **quantity/value is associated to each cell**
- Example: Ligo/Virgo Multi-Order probability Map



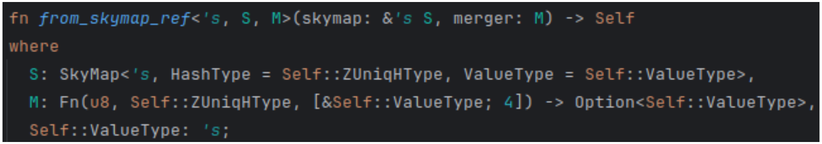


MOM

- Origin
  - Martinez-Castellanos, Singer et al. 2021: *“Multi-Resolution HEALPix Maps for Multi-Wavelength and Multi-Messenger Astronomy”*
    - C++ implementation in mhealpy
  - Now, Multi-Order Sky Maps
- Similar idea at CDS: *“Valued-MOC”*, now **MOM**
  - BMOC (<2018) in CDS HEALPix (Java and Rust)
    - particular MOM with value = boolean
    - single bit added to the `ZUNIQ` numbering
    - operations dedicated to boolean case, no genericity
  - MOM implementation in CDS HEALPix Rust (since 09/2024)
    - very generic (how it compares with `mhealpy`?)
- Examples
  - HATS structure: merge siblings while `#src <1M`
    - see also Rust/Python code by K. Malanchev
  - **New!** χ^2 MOM
  - **Idea**: x-match spurious association (and proba) maps

MOMs in CDS HEALPix Rust

- MOM creation from a skymap is very generic
  - type of index is generic (`u32`, `u64`, ...)
  - type of value is generic (`f32`, `i64`, list, ...)
  - function merging (or not) siblings is generic
    - takes parent cell `order` and `ipix`
    - takes the 4 siblings values
    - return either the merged value or `None`
  - single requirement from the skymap: iterate on `(ipix, value)` tuples on ascending `ipix` order



```rust
fn from_skymap_ref<'s, S, M>(skymap: &'s S, merger: M) -> Self
where
  S: SkyMap<'s, HashType = Self::ZUniqHType, ValueType = Self::ValueType>,
  M: Fn(u8, Self::ZUniqHType, [&Self::ValueType; 4]) -> Option<Self::ValueType>,
  Self::ValueType: 's;
```

2 examples of MOM creation from Skymaps with hpx-cli

χ^2 MOMs

- Build a χ^2 MOM from a density/count Skymap
  - χ^2-test: are 4 siblings from a same Poisson distribution?
    - yes: merge into the parent cell
    - no: do not merge the cells
- Example with hpx-cli
- Input file `gaia_edr3_dist.csv`
  - 1 467 744 819 rows, i.e. almost **1.5 billion rows**
  - 10 columns
  - **165 GB**

```
> head -10 gaia_edr3_dist.csv
source_id,RA_ICRS,DE_ICRS,r_med_geo,r_lo_geo,...
10000000573220000000,104.87975539563,55.95486459306,784.559
10000001217464727004,104.85627575018,55.96825004364,1875.965
10000001561062200032,104.86295428908,55.97882280840,2708.93
10000001561062214440,104.85273193584,55.98099099003,1336.845
...
```

χ^2 MOMs

- Density map FITS file creation at order 11

```
> time cut -f , -d 2,3 gaia_edr3_dist.csv \
  | tail -n +2 \
  | hpx map dens 11 gaia_edr3_dist.dens.fits list -d ,

real    8m25,812s (=> 340 MB/s or 2.9e+6 src/s)
user    11m7,242s
sys     3m40,669s
```

- PNG creation from the **385 MB skymap** FITS file

```
> time hpx map view --silent gaia_edr3_dist.dens.fits \
    gaia_edr3_dist.dens.png allsky 400

real    0m20,361s (14s on my laptop)
user    0m19,246s
sys     0m0,492s
```

χ^2 MOMs

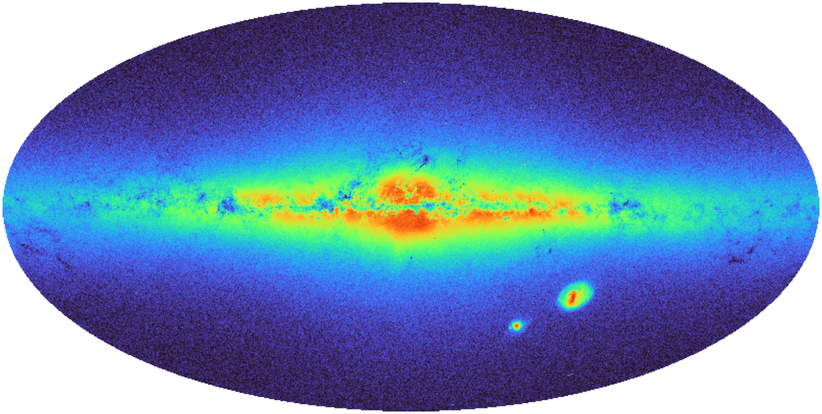


Figure 2: Skymap at order 11, 1.5 billion sources

χ^2 MOMs

- Create a χ^2 MOM from the Skymap
  - **from 385 MB to 13 MB in 1s**

```
> time hpx map convert gaia_edr3_dist.dens.fits \
    gaia_edr3_dist.dens.mom.fits dens2chi2mom

real    0m0,978s
user    0m0,661s
sys     0m0,317s
```

- PNG creation from the 13 MB FITS file

```
> time hpx mom view --silent gaia_edr3_dist.dens.mom.fits
    gaia_edr3_dist.dens.mom.png allsky 400

real    0m0,494s
user    0m0,478s
sys     0m0,013s
```

χ^2 MOMs

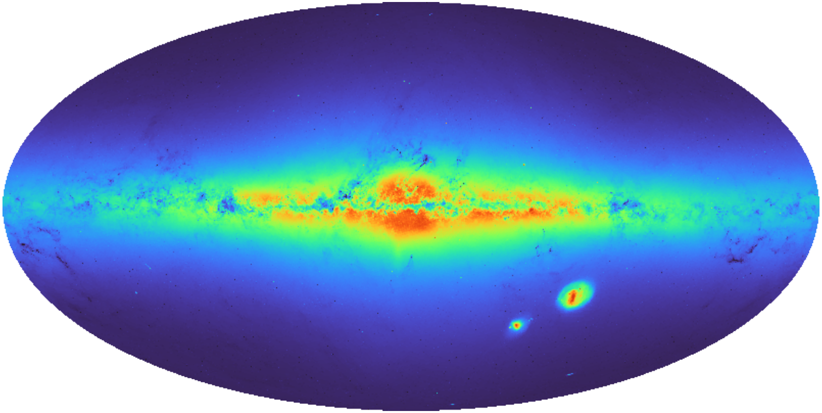


Figure 3: χ^2 MOM

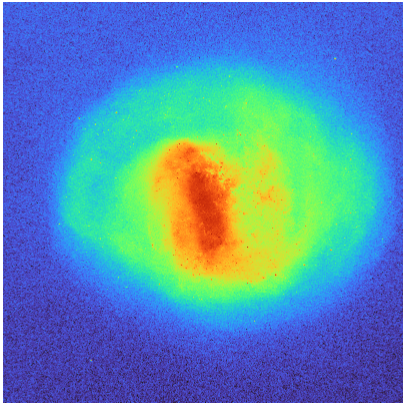
χ^2 MOMs

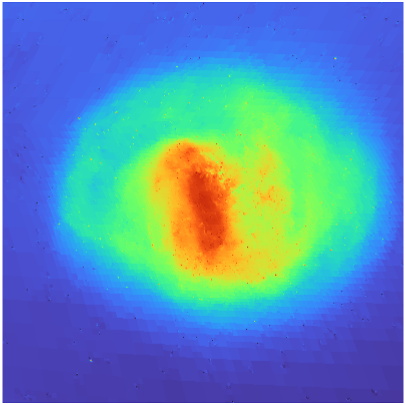
Zoom around the LMC

```
> hpx map view --silent gaia_edr3_dist.dens.fits \
    gaia_edr3_dist.dens.lmc.png \
    custom -l 280.4652 -b -32.8884 \
    sin 400 400 \
    --x-bounds [-0.15..0.15] --y-bounds [-0.15..0.15]

> hpx mom view --silent gaia_edr3_dist.dens.mom.fits
    gaia_edr3_dist.dens.mom.lmc.png \
    custom -l 280.4652 -b -32.8884 \
    sin 400 400 \
    --x-bounds [-0.15..0.15] --y-bounds [-0.15..0.15]
```

χ^2 MOMs





Left: skymap. Right: $\chi^2 MOM$ (we see quantization due to merged cells in low density regions; signal in high density regions preserved; horizontal alignement due to galactic projection)

χ^2 MOMs

- On the previous example:
  - 385 vs 13 MB: **x30** compression
  - PNG creation 20 vs 0.5 s: **x40** faster
  - **White noise removed** on a statistical basis (χ^2)!
  - While keeping high resolution were density varies quickly (order 11 $\sim= 1.7' \times 1.7'$)

HATS struct MOM

Instructions to get HATS list of cells (down-top approach) for all VizieR catalogues (here for Gaia DR3):

- download Skymap file (T. Boch) at order 10 ($3.4'x3.4'$)
- create a MOM from the Skymap with hpx-cli
- view with hpx-cli
- convert the FITS MOM into FITS BINTABLE with hpx-cli
- load BINTABLE FITS in TOPCAT

HATS struct MOM

Command lines:

```
> wget -O gdr3.d10.skymap.fits "${URL}"
> hpx map convert gdr3.d10.skymap.fits gdr3.hats.fits \
      count2mom 1000000
> hpx mom view gdr3.hats.fits gdr3.hats.png \
      -c linear allsky 800
> hpx mom convert gdr3.hats.fits gdr3.hats.bintable.fits \
      bintable
> topcat gdr3.hats.bintable.fits &
```

HATS struct MOM

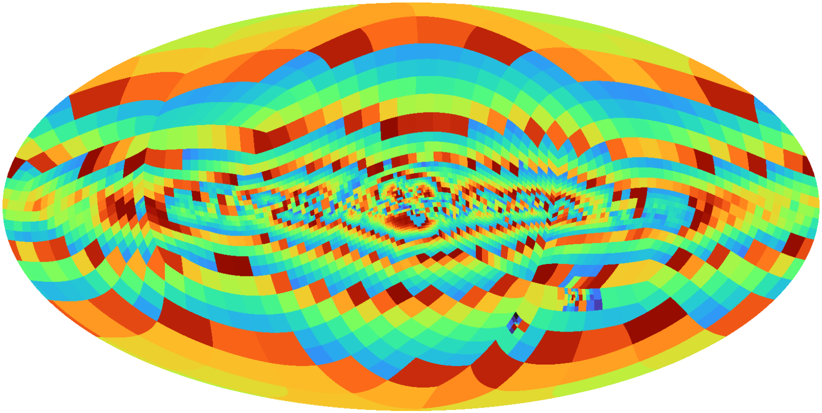
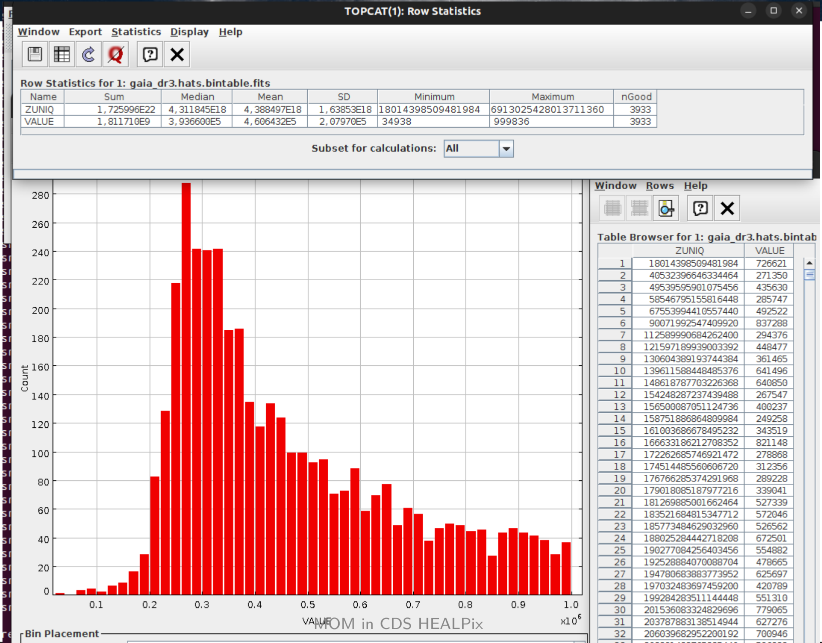


Figure 4: Number of rows per HEALPix cell in Gaia DR3 HATS structure (max 1 million rows per cell)

HATS struct MOM



TOPCAT(1): Row Statistics

Row Statistics for 1: gaia_dr3.hats.bintable.fits

Name	Sum	Median	Mean	SD	Minimum	Maximum	nGood
ZUNIQ	1,725996E22	4,311845E18	4,388497E18	1,63853E18	18014398509481984	6913025428013711360	3933
VALUE	1,811710E9	3,936600E5	4,606432E5	2,07970E5	34938	999836	3933

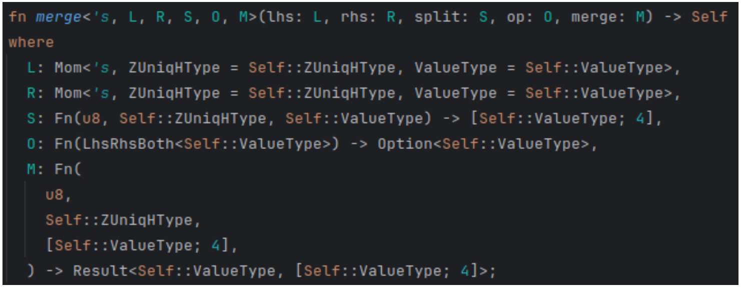
Subset for calculations: All

Table Browser for 1: gaia_dr3.hats.bintab

	ZUNIQ	VALUE
1	18014398509481984	726621
2	40532396646334464	271350
3	49539595901075456	435630
4	58546795155816448	285747
5	67553994410557440	492522
6	90071992547409920	837288
7	112589990684262400	294376
8	121597189939003392	448477
9	130604389193744384	361465
10	139611588448485376	641496
11	148618787703226368	640850
12	154248287237439488	267547
13	156500087051124736	400237
14	158751886864809984	249258
15	161003686678495232	343519
16	166633186212708352	821148
17	172262685746921472	278868
18	174514485560606720	312356
19	176766285374291968	289228
20	179018085187977216	339041
21	181269885001662464	527339
22	183521684815347712	572046
23	185773484629032960	526562
24	188025284442718208	672501
25	190277084256403456	554882
26	192528884070088704	478665
27	194780683883773952	625697
28	197032483697459200	420789
29	199284283511144448	551310
30	201536083324829696	779065
31	203787883138514944	627276
32	206039682952200192	700946

Operation between 2 MOMs

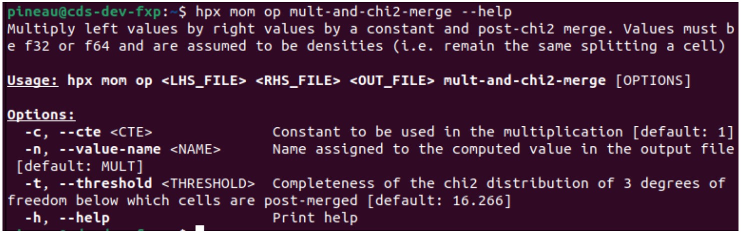
- Operation between 2 MOMs also **very generic**, requires:
  - a split function to split a parent cell into 4 siblings
  - an operation returning an optional value from either:
    - one value from the left MOM
    - one value from the right MOM
    - one value from both MOMs
  - a possible post-merge operation



```rust
fn merge<'s, L, R, S, O, M>(lhs: L, rhs: R, split: S, op: O, merge: M) -> Self
where
  L: Mom<'s, ZUniqHType = Self::ZUniqHType, ValueType = Self::ValueType>,
  R: Mom<'s, ZUniqHType = Self::ZUniqHType, ValueType = Self::ValueType>,
  S: Fn(u8, Self::ZUniqHType, Self::ValueType) -> [Self::ValueType; 4],
  O: Fn(LhsRhsBoth<Self::ValueType>) -> Option<Self::ValueType>,
  M: Fn(
    u8,
    Self::ZUniqHType,
    [Self::ValueType; 4],
  ) -> Result<Self::ValueType, [Self::ValueType; 4]>;
```

Operation between 2 MOMs

- Examples
  - merge 2 probability MOMs
    - implemented in Ligo/Virgo (`mhealpy`) I guess
    - not implemented (yet?) in CDS HEALPix Rust
    - a few lines to code `split`, op and `merge`
  - rate of spurious x-match
    - Density MOM 1 times density MOM 2 time constant value (cone-search area)
    - post-merge χ^2 compatible number of spurious matches
    - (then merge with XMatch result density MOM to compute xmatch prior MOM)



```
pineau@cds-dev-fxp:~$ hpx mom op mult-and-chi2-merge --help
Multiply left values by right values by a constant and post-chi2 merge. Values must b
e f32 or f64 and are assumed to be densities (i.e. remain the same splitting a cell)

Usage: hpx mom op <LHS_FILE> <RHS_FILE> <OUT_FILE> mult-and-chi2-merge [OPTIONS]

Options:
  -c, --cte <CTE>              Constant to be used in the multiplication [default: 1]
  -n, --value-name <NAME>      Name assigned to the computed value in the output file
 [default: MULT]
  -t, --threshold <THRESHOLD>  Completeness of the chi2 distribution of 3 degrees of
freedom below which cells are post-merged [default: 16.266]
  -h, --help                   Print help
```

The question of the representation/serialization.

MOC Representation

Evolution of the MOC internal representation and FITS serialisation:

- MOC 1.0: **by layer** representation
  - Java library: one `IPIX` array per `ORDER`
  - FITS: sorted list of `NUNIQ`
    - low resolution cells before high resolution cells
    - **no `NUNIQ` for Time (v2.0) and Frequency (v2.1)**
- MOC 2.0: **max order range** representation
  - both in memory and in FITS
  - **faster** set operations (not, and, or, . . .)
  - operation possible on **streamed** (large) MOCs with a **tiny memory footprint**
- v2.0 sometimes more compact than v1.0, not always
  - x2 in favor of v1.0 in case of isolated cells

SUNIQ numbering

NUNIQ versus SUNIQ

- NUNIQ: described in the **HEALPix paper**
  - base cell $\in [0, 11]$ coded on 4 bits ($[0, 15]$)
  - NUNIQ: add 4 to the base cell number
    - no extra bit required
  - $NUNIQ = (4 << (ORDER << 1)) + IPIX$
- **But**, no extra value available in bit patterns of **Time or Frequency** cells
- SUNIQ: similar to NUNIQ, uses a **sentinel bit**
  - marks the limit of the highest possible value
  - SUNIQ add 16 to the base cell number
    - requires an extra bit
  - $SUNIQ = (16 << (ORDER << 1)) | IPIX$
- SUNIQ solves the problem of new MOC quantities!
- **But** SUNIQ **natural ordering still** based first on ORDER, then IPIX (so **layer by layer**)

MOM representation

- MOM
  - each cell as a distinct value thus, contrary to MOCs, **succesive cells cannot be merged into ranges**
  - **efficient/stream operations need cells to follow the Z-Order curve!!**
- How to add the `ORDER` information in the `IPIX`, preserving the max res ordering?
  - **sentinel bit on the LSB** part instead of on the MSB part!
  - adopted since BMOC origine, also in google s2geometry (someone pointed this to me 3 months ago)
- Here the magic `ZUNIQ` formula:

$$ZUNIQ = ((IPIX << 1)|1) << ((ORDMAX - ORDER) << 1)$$

- `ZUNIQ` allows for **efficient operations**, on **streamed** (i.e. possibliy **large the RAM**) **MOMs**, with a **tiny memory footprint**

Unified view

- Today:
  - MOC v1.0: sorted list of NUNIQ
  - MOC v2.0: sorted list of MAX ORDER ranges
  - BMOC: sorted list of **ZUNIQ** with extra bit flag
  - MOM:
    - Ligo/Virgo: sorted list of NUNIQ, VALUES tuples
    - CDS HEALPix: sorted list of **ZUNIQ**, VALUE tuples
- But:
  - NUNIQ valid only for HEALPIX
    - MOM could be generalized to **Time and Frequency**
  - **NUNIQ not adapted for**
    - efficient operations
    - streaming operations (large MOMs)
- Use **ZUNIQ** everywhere?!

Unified view

MOC is ia particular BMOC, BMOC is a particular MOM (Multi Order SkyMap), so what about:

- MOC: BINTABLE with ordered ZUNIQ column
  - no additional columns
- BMOC: BINTABLE with ordered ZUNIQ column
  - plus extra boolean column
- MOM: BINTABLE with ordered ZUNIQ column
  - plus extra column(s) (possibly pointing to variable length data)

Wrap-up slide

- MOM recently implemented in CDS HEALPix Rust
- Very generic implementation
  - core algo implemented
  - custom needs: implement basic functions
- A few use case examples:
  - merge probability MOMs
  - χ^2 density MOMs
  - compute x-match probabilities
  - ... any use case from you!
- FITS serialisation
  - `mHEALPy`: NUNIQ (internal representation??)
  - CDS HEALPix: `ZUNIQ`
- **I favor `ZUNIQ` for MOMs**, but
  - also for BMOCs (with a separate flag)?
  - also for MOCs?
- A short note about `SUNIQ` and `ZUNIQ`?