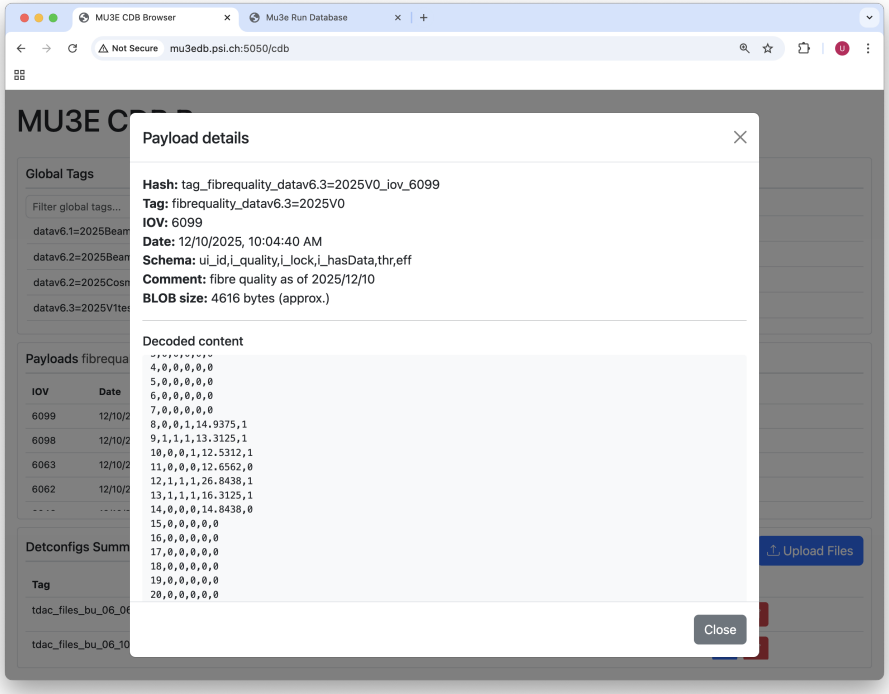
CDB: calFibreQuality

- calFibreQuality
  - ▷ indexed by ASIC (active in 2025: 8, . . . , 15)
    - different from tiles (are indexed by channel)
  - ▷ includes (per ASIC)
    - has_data
    - lock
    - threshold
  - ▷ payload contents from Marius, 2+95 iovs (runs 3481 . . . 6099)
  - ▷ included in datav6.3=2025V0

⇒ Somebody should put this to use in mu3eTrirec or elsewhere :-)



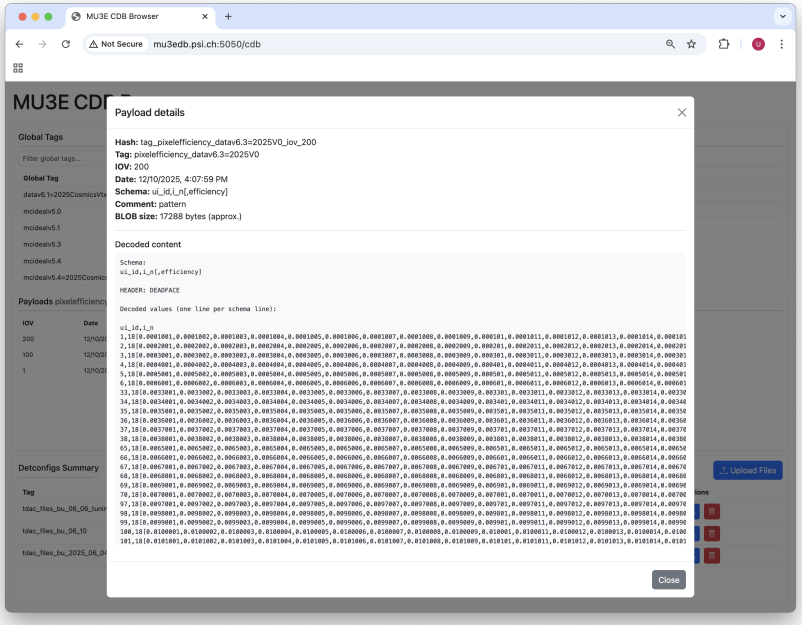
- Accessors

```
// -- direct accessors
Status getAsicStatus(uint32_t asicID);
bool   getAsicLock(uint32_t asicID);
bool   getAsicHasData(uint32_t asicID);
int    getAsicQuality(uint32_t asicID);
double getAsicThreshold(uint32_t asicID);
double getAsicEfficiency(uint32_t asicID);
```

CDB: calPixelEfficiency

- calPixelEfficiency
  - thanks to Haris for preparatory work
  - indexed by global chipID
  - n efficiencies for the entire sensor (n is not fixed for indiscrimate access)
  - 3 payloads unphysical numbers (1., 0., pattern)
  - included in datav6.3=2025V0
- To be filled with real numbers
  - ideally with some run dependence
    (else you don't really need the CDB :-)
- Accessors
  - possibly more descriptive :-)

```
// -- direct accessors
double efficiency(uint32_t id,
                  int i/*i = 0 .. 17*/);
```



cdbWritePayload

- mu3eUtil/cdb/test/cdbWritePayload.cpp
  - ▷ can read directly from alignment/sensors
  - ▷ can read directly from alignment/mppcs
  - → rest will follow (eventually)
  - ▷ Usage example

```
./bin/cdbWritePayload -c mppcalignment -g testdevroot -j ~/data/mu3e/cdb/ \
-f ~/mu3e/software/mu3e/run/mu3e_alignment.root
```

see code for documentation

(In case it's not obvious . . .)

- Access to and usage of CDB "cal" classes

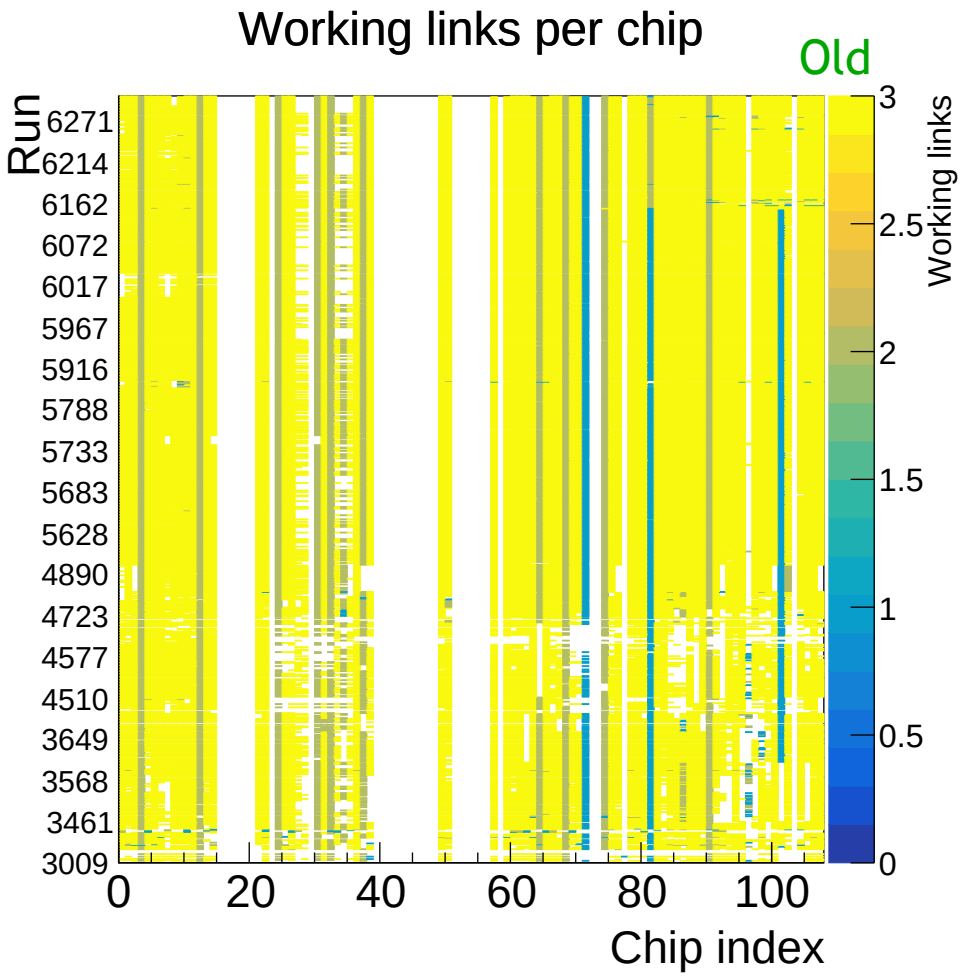
```
// from mu3eTrirec//src/mu3e/rec/SiDet.cpp
    #include "mu3e/tools/cdb.h" //???
    if(auto cal = mu3e::cdbCal<calPixelAlignment>("pixelalignment_")) {
        uint32_t calId = 0;
        cal->resetIterator();
        while(cal->getNextID(calId)) {
            entry.vars.clear();
            entry.id = cal->id(calId);
            entry.vx = cal->vx(calId);
            entry.vy = cal->vy(calId);
.. snip ..


// from mu3eSim/sort/mu3eSort.cpp
    calPixelQualityLM* midPixelQuality = mu3e::cdbCal<calPixelQualityLM>("pixelqualitylm_");

.. snip ..

    if(midPixelQuality != nullptr
        && calPixelQualityLM::Good !=
           midPixelQuality->getLinkStatus(hitId.sensorId.value(), hitId.linkId())
        ) {
            if(mu3e::conf.verbose >= 3) {
                MU3E_WARN("skip non-good link hit 0x%08X, link =
                          %d\n", hitId.value(), hitId.linkId());
            }
            continue;
        }
.. snip ..
```

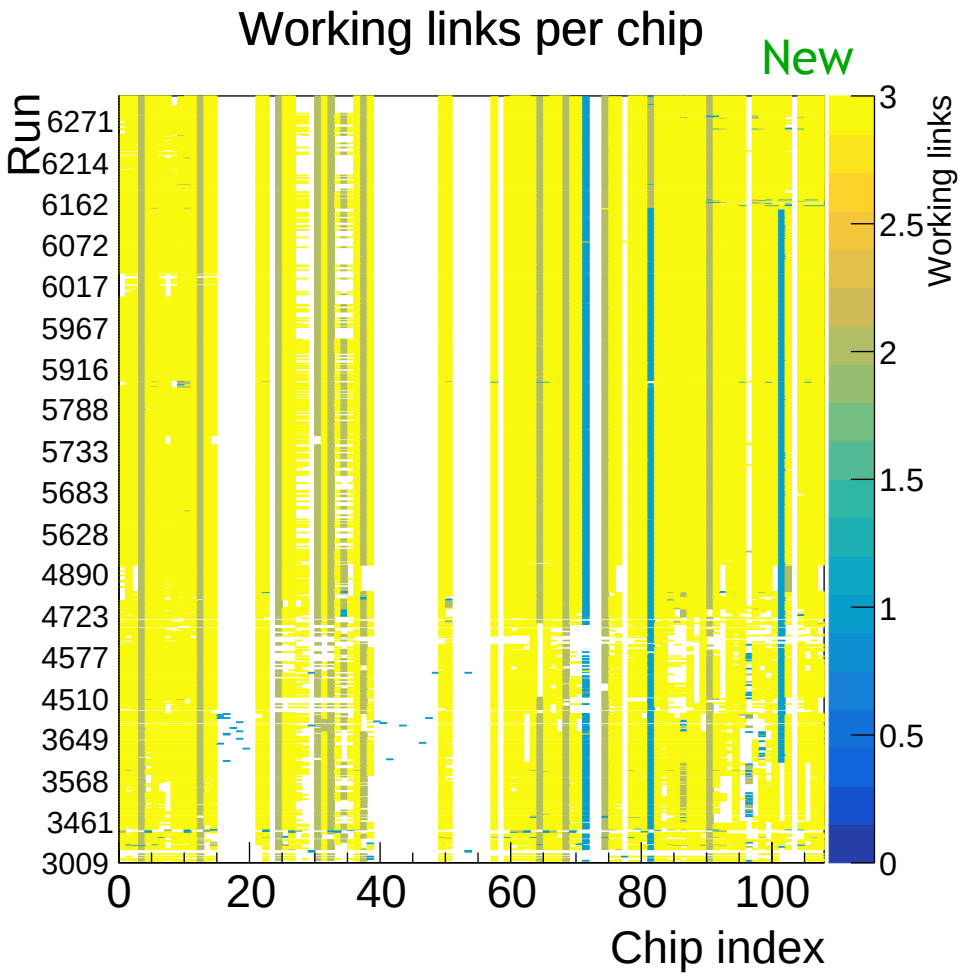
- If desired, a python api/interface to the CDB is likely very easy

Pixel quality

- Starting point (TS): Ignore the "E" label
  - ▷ only look at the LVDS error rate
  - → only very minor changes

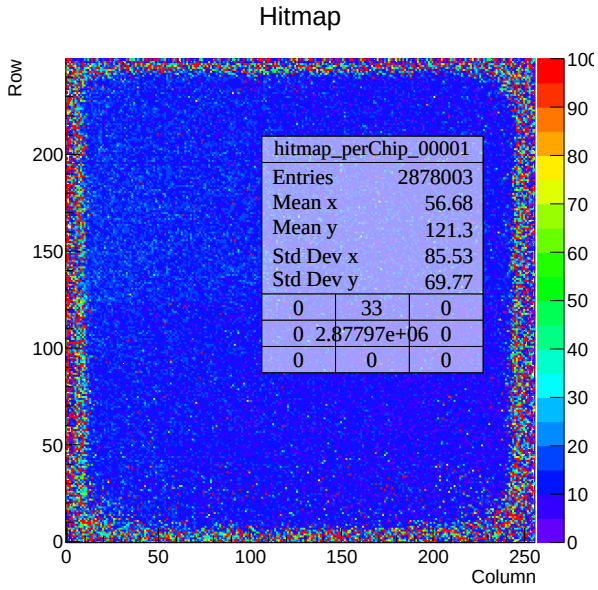
    (if you worry about LVDS errors, you can extract the hitmap overflow [rate] for yet another handle)





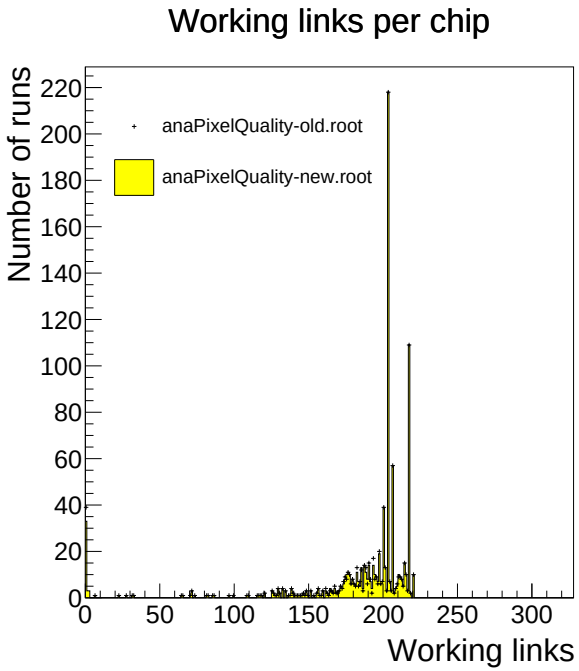
Pixel Quality II

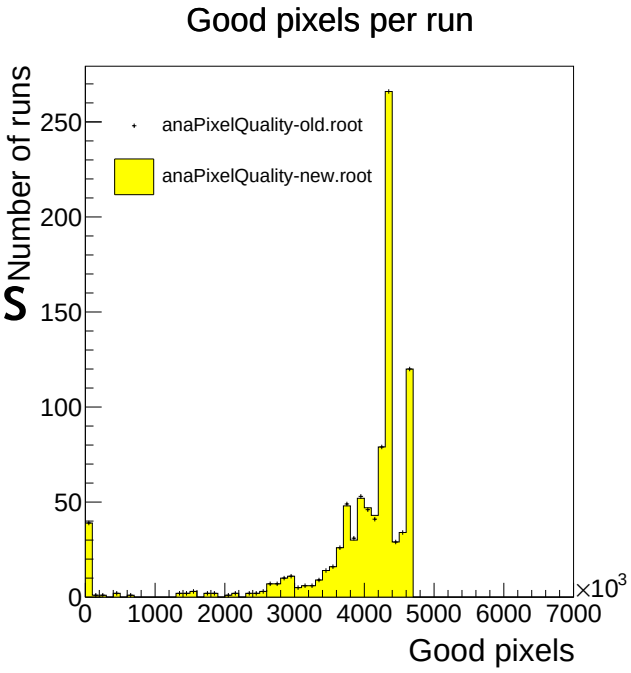
- Deployed to all instances
  - replaces the (old) datav6.3=2025V0 payloads
- hitmap overflows (random example):



- r = #overflow hits / #hits with legal addresses
  - store inverse in uint_32 "ui_linkM"
  - if $r > 0.1$ rounding inaccuracies
  - accessor

```
double getLVDSOverflowRate(unsigned int chipid);
```





RDB: Changes to significant run list

- No changes
  - all tiles and fibres good runs are also significant runs
- Jak's "good run" list for vertex updated in RDB
  - now contains

```
6116,5102,4900,4899,4898,4897,4896,4894,4893,4892,4891,4890,4889,4888,
4887,4886,4885,4884,4883,4882,4881,4880,4878,4877,4876,4873,4872,4871,
4870,4869,4868,4866,4865,4864,4863,4758,4757,4756,4748,4746,4745,4744,
4743,4742,4740,4739,4728,4727,4726,4725,4724,3822,3820,3819,3818,3817,
3816,3815,3814,3813,3812,3799,3798,3797,3796,3795,3794,3657,3653,3652,
3651,3649,3646,3645,3644,3642,3638,3637,3636,3635,3634,3633,3629,3627,
3625,3624,3622,3620,3619,3606,3604,3603,3602,3586,3585,3583,3582,3581,
3578,3576,3575,3574,3568,3514,3504,3503,3502,3501,3500,3499,3498,3497,
3496,3495,3494,3493,3492,3488,3487,3475,3474,3472,3471,3416,3312
```

  - still no overlap with tiles or fibres
  - 48 runs overlap if you select "Not Bad" for vertex
    ("bad": less than 1M good pixels or more than 40 chips with 3 bad links)

RDB - pagination

- RDB issue with large number of runs "Max runs (limit)"

⇒ pagination