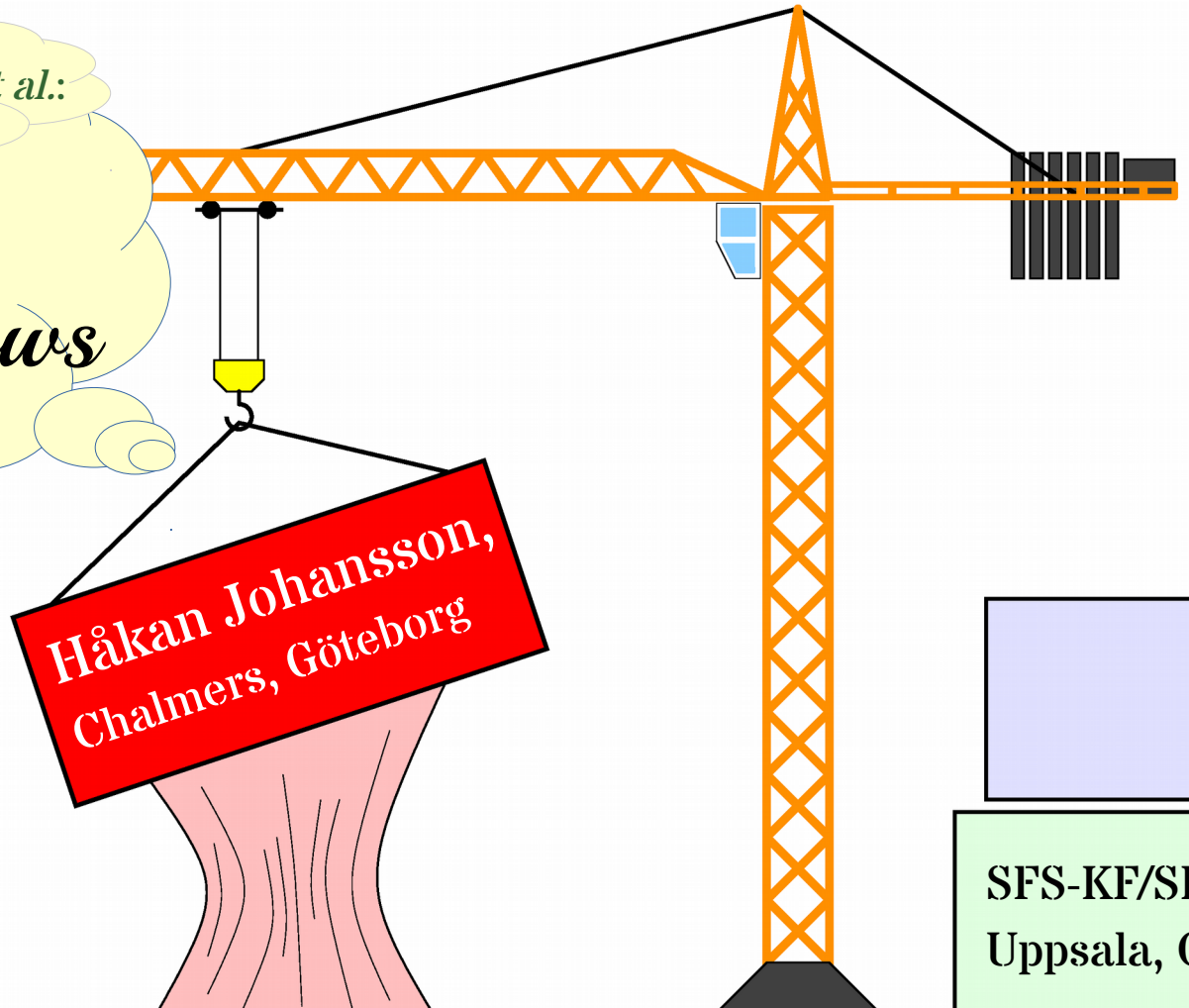
From sticky analysis...

with Michael Munch *et al.*:

...to fast VME shadows

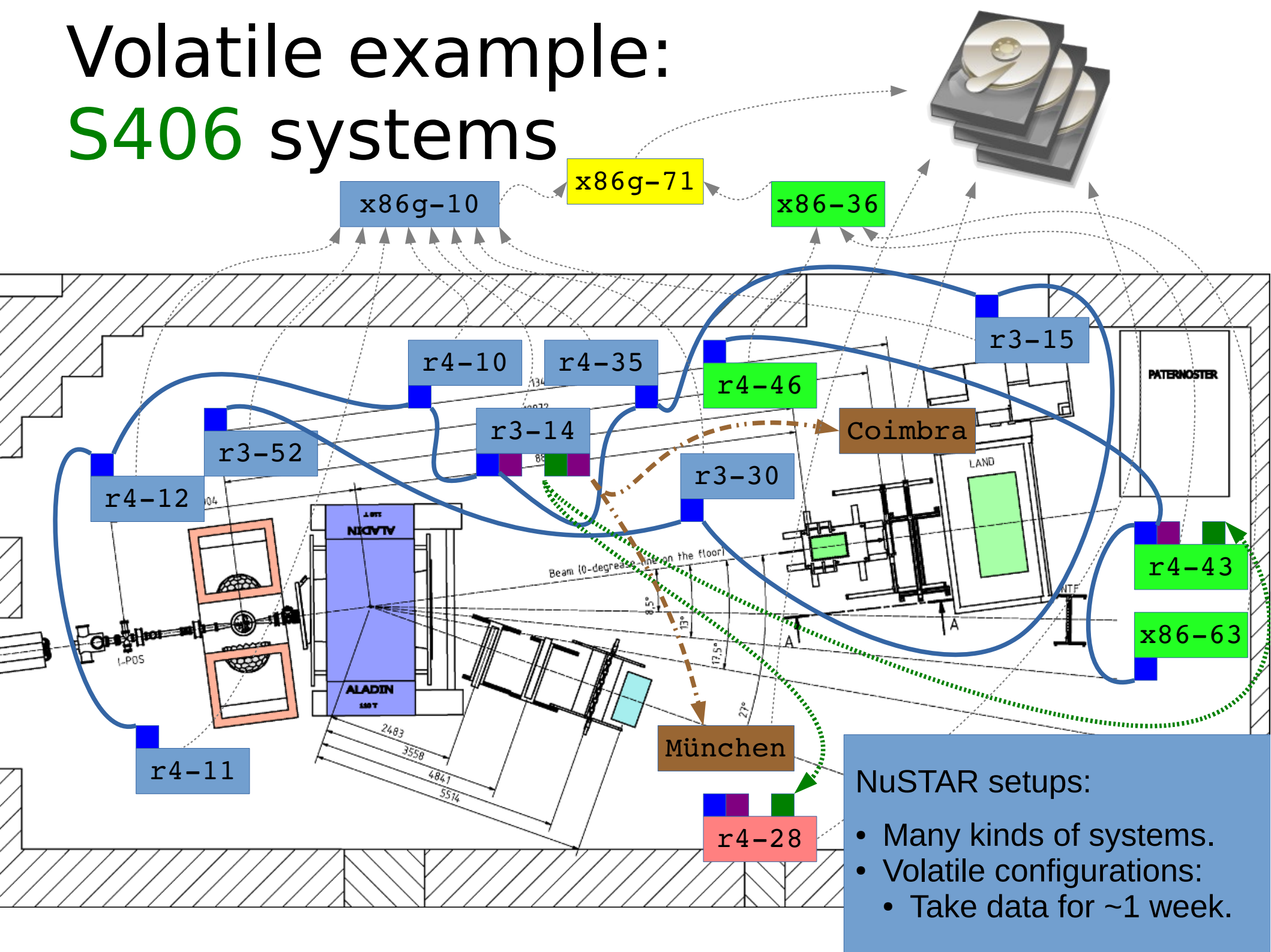
→ page 13



Håkan Johansson,
Chalmers, Göteborg

SFS-KF/SFAIR,
Uppsala, Oct, 2018

Volatile example:
S406 systems



NuSTAR setups:

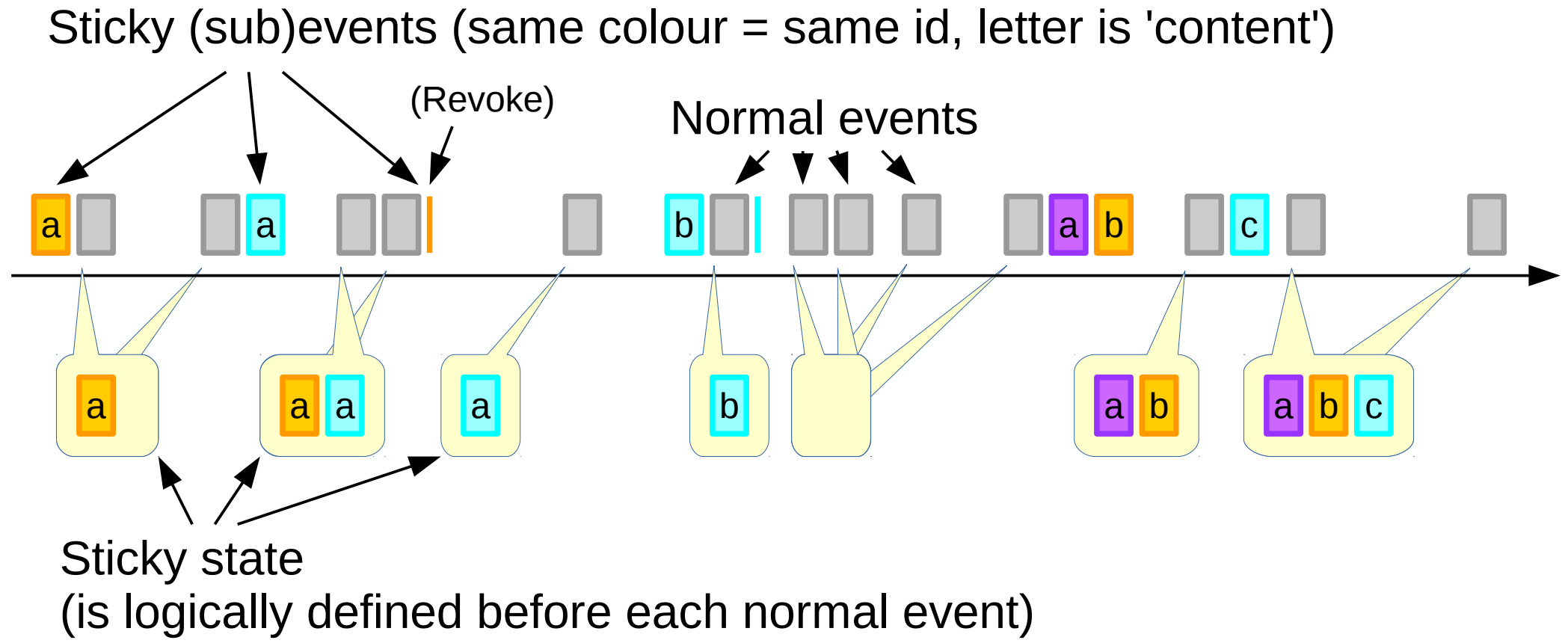
- Many kinds of systems.
- Volatile configurations:
  - Take data for ~1 week.

Why sticky events?

- Store slow-control information (HV settings, magnet currents...) in the data stream
  - Integrate with a distributed DAQ (NuSTAR).
  - Follow the DAQ topology.
- 'Normal' events not suitable:
  - Just flow through the DAQ / analysis.
  - Late connected clients / files would not get earlier set values.
- Concept: sticky events.
  - Delivered however late the connection is.

Sticky events: simple semantics

- Sticky subevents are valid until replaced
- … or revoked (replace by nothing)

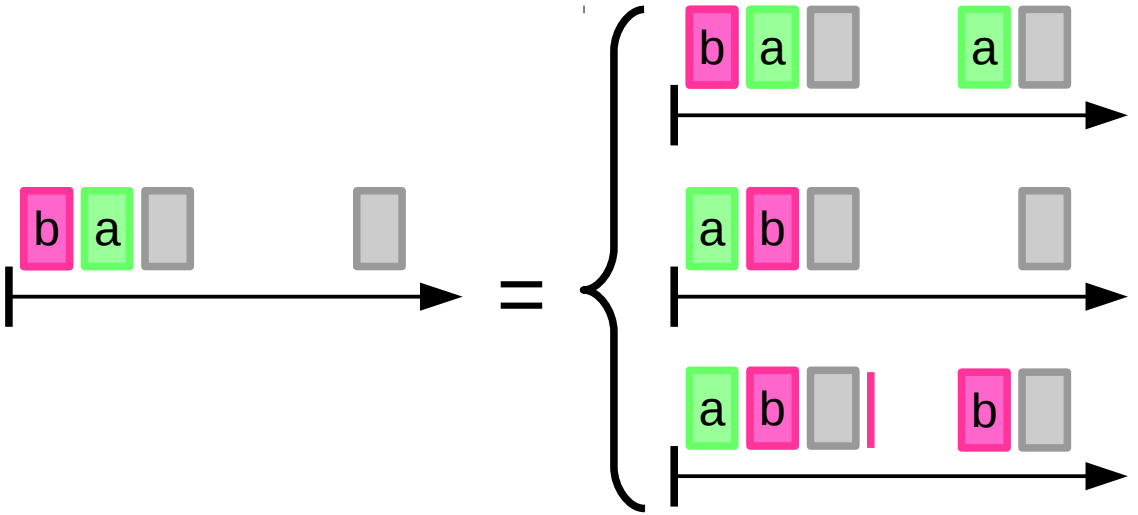


Guaranteed delivery

- A receiver (either file or network client) will **before** each normal event have received *exactly* the (at that point) active set of sticky subevents.

Sticky subevents may be delivered:

- Multiple times.
- In any order.

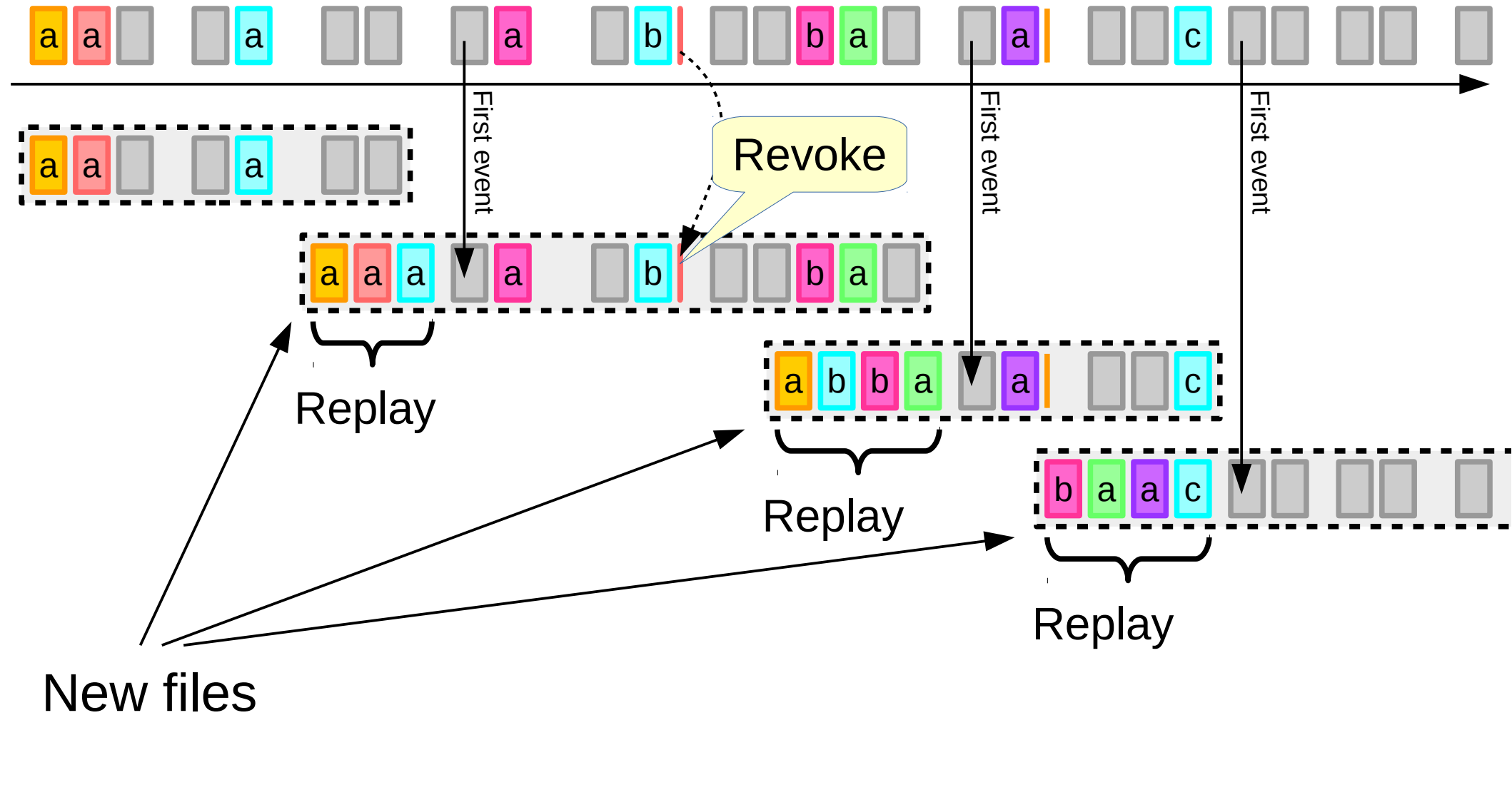


DAQ / proxy servers

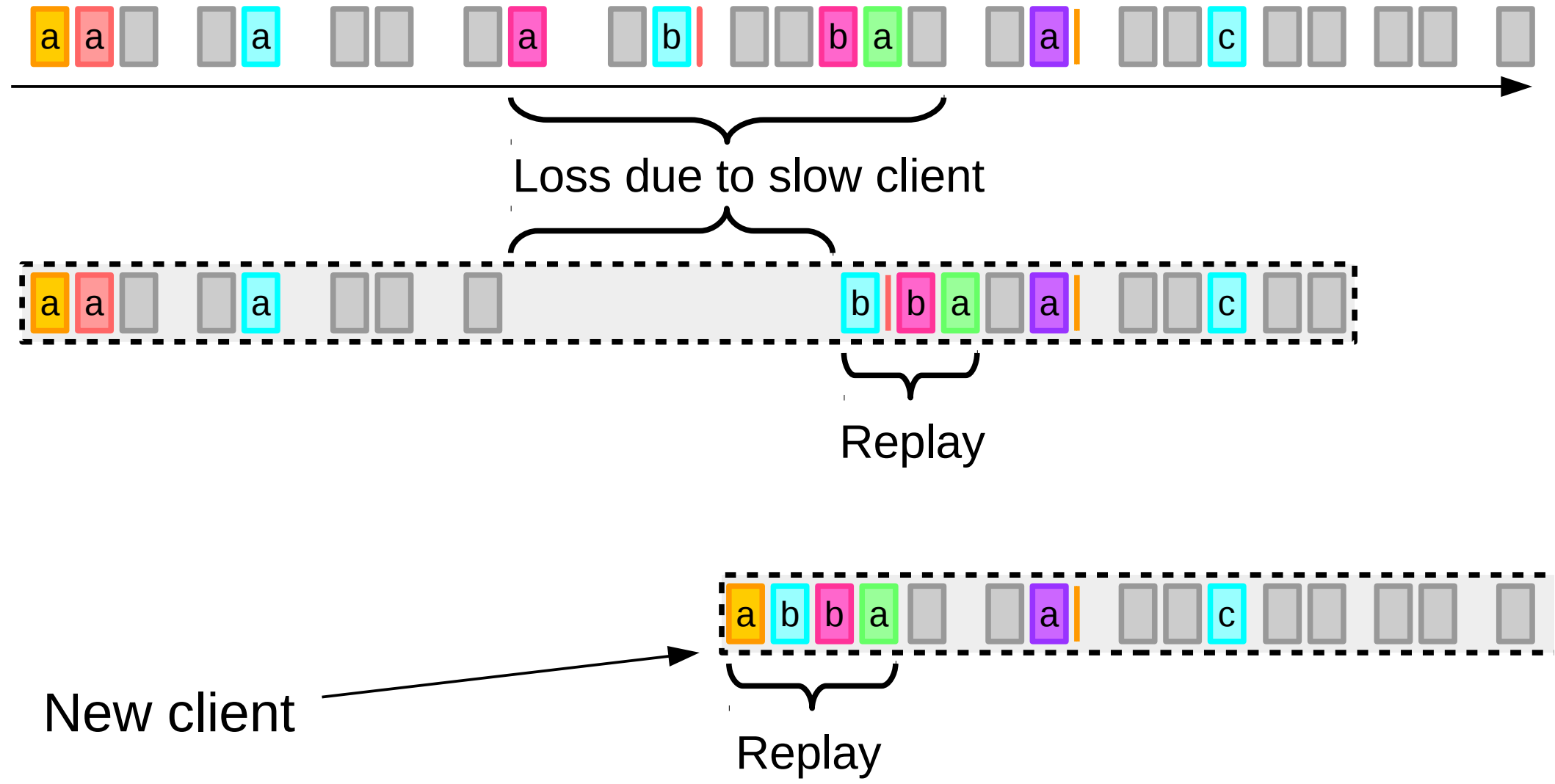
- Absorb the complications in standard programs.
- Keeps analysis clients simple.

Not so much a design choice,
rather a lucky side-effect.

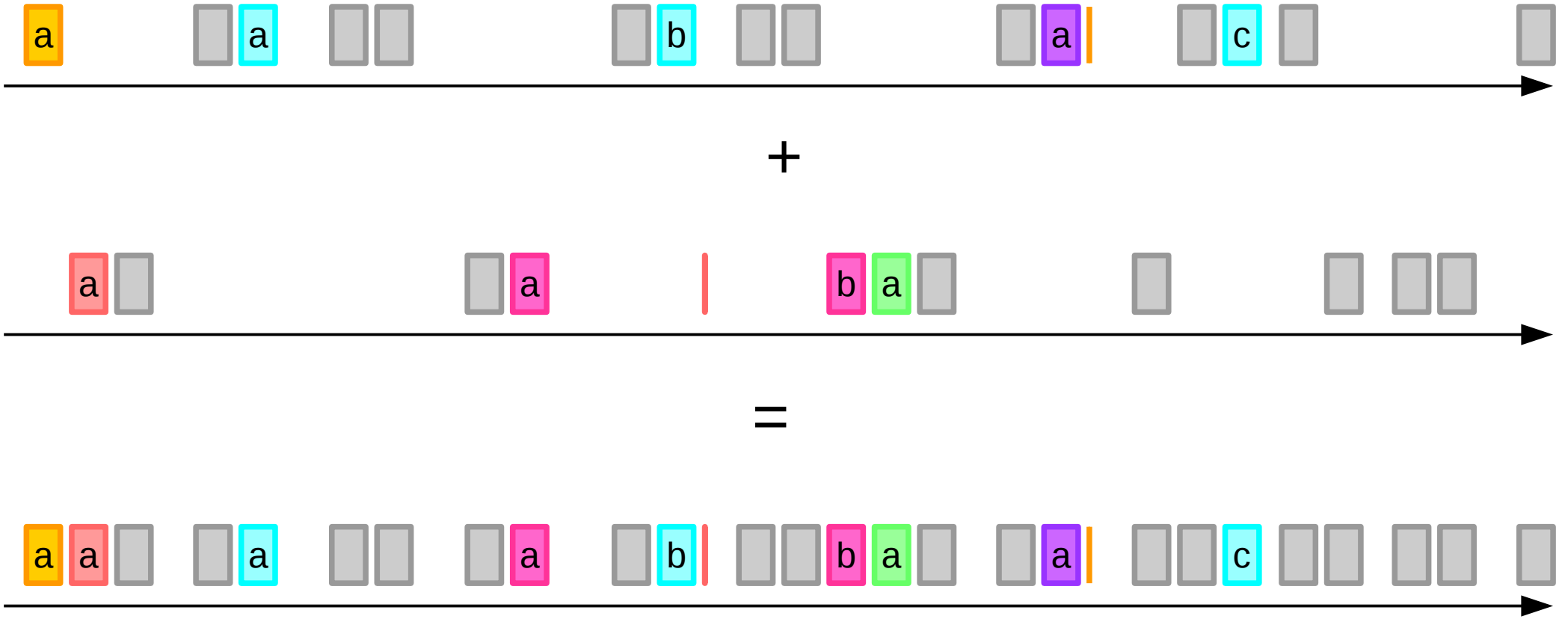
Output stages keep track



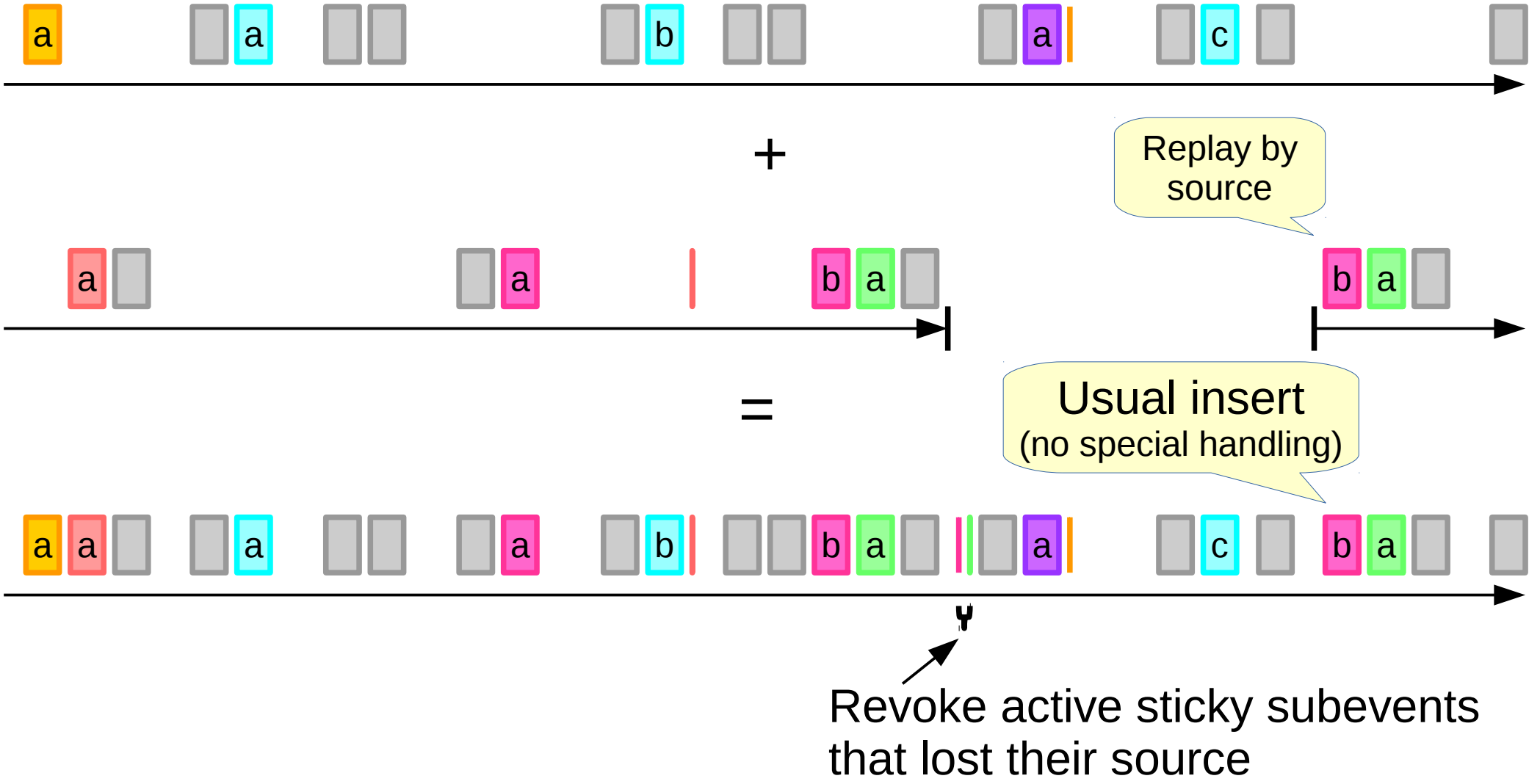
Output stages keep track (network)



Merging / time sorting



Merging – loss of source



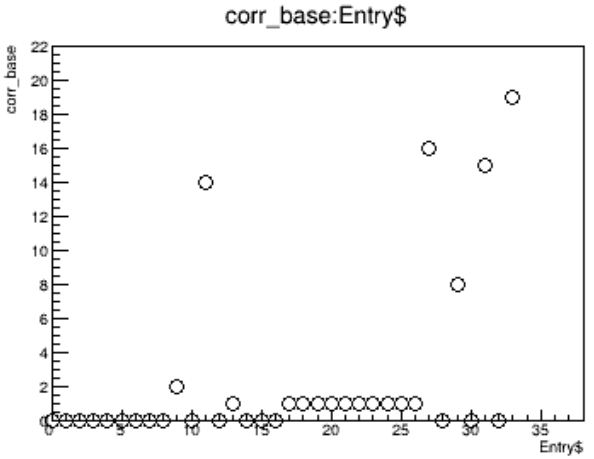
Correlating sticky events, with

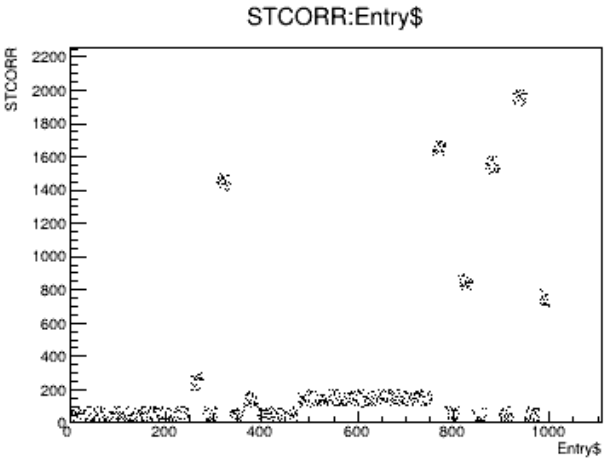
`corr_base` – In the sticky events, which are set once in a while, `random: 0-19`.
`STCORR` – In the normal events, set as `corr_base * 10 + (random: 0-9)`

Root file has two trees:

- h101 – Normal events
- hsticky – Sticky events

```
hsticky->("corr_base:Entry$");
```

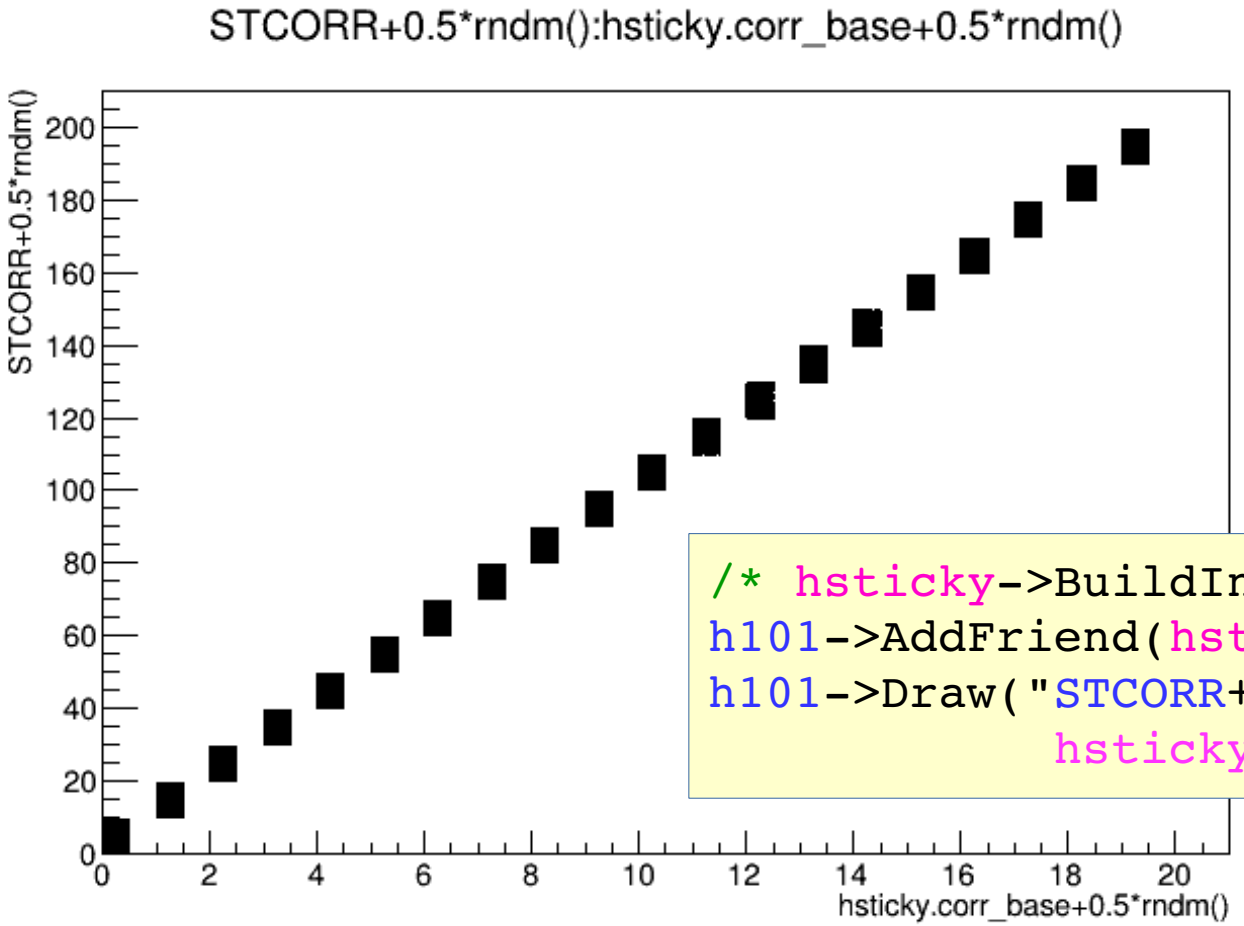





```
h101->Draw("STCORR:Entry$");
```

Correlating sticky events, with

`corr_base` – In the sticky events, which are set once in a while, `random: 0-19`.
`STCORR` – In the normal events, set as `corr_base * 10 + (random: 0-9)`



Root file has two trees:

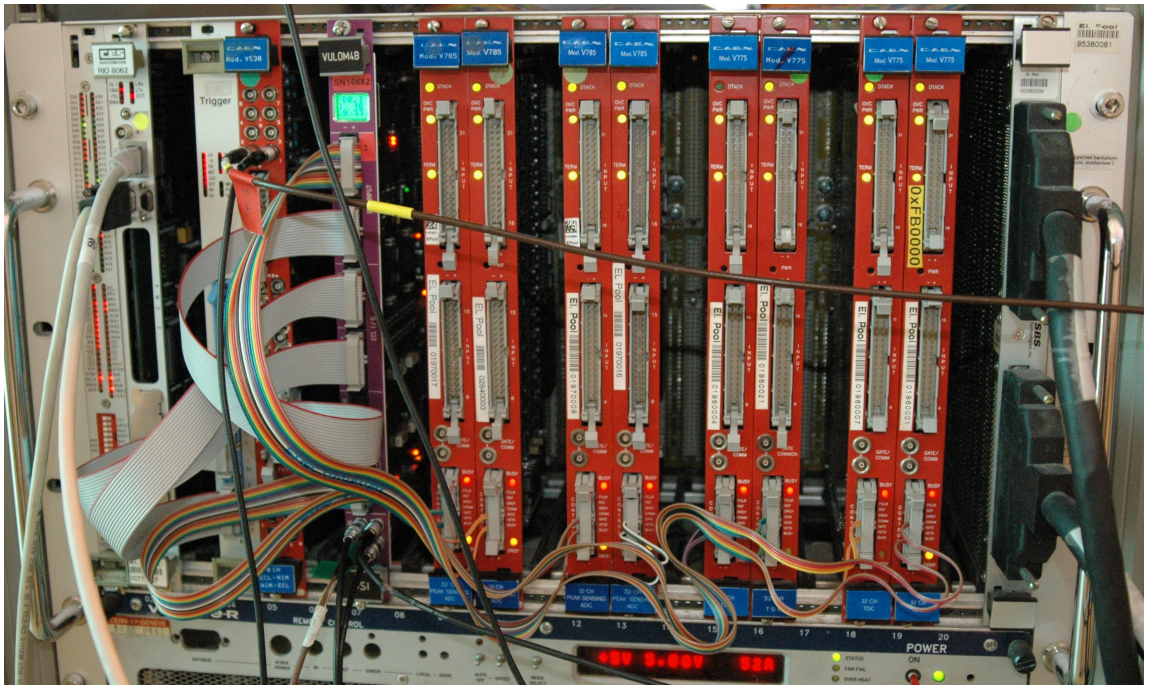
- h101 – Normal events
- hsticky – Sticky events

STIDX – sticky event index (in both trees)

```
/* hsticky->BuildIndex("STIDX"); */
h101->AddFriend(hsticky);
h101->Draw("STCORR+0.5*rndm():
            hsticky.corr_base+0.5*rndm()");
```

Fast VME shadows

M. Munch, J. H. Jensen,
B. Löher, H. Törnqvist, and H. T. Johansson



arXiv:1810.03574

How about...

?

How about...

More statistics?

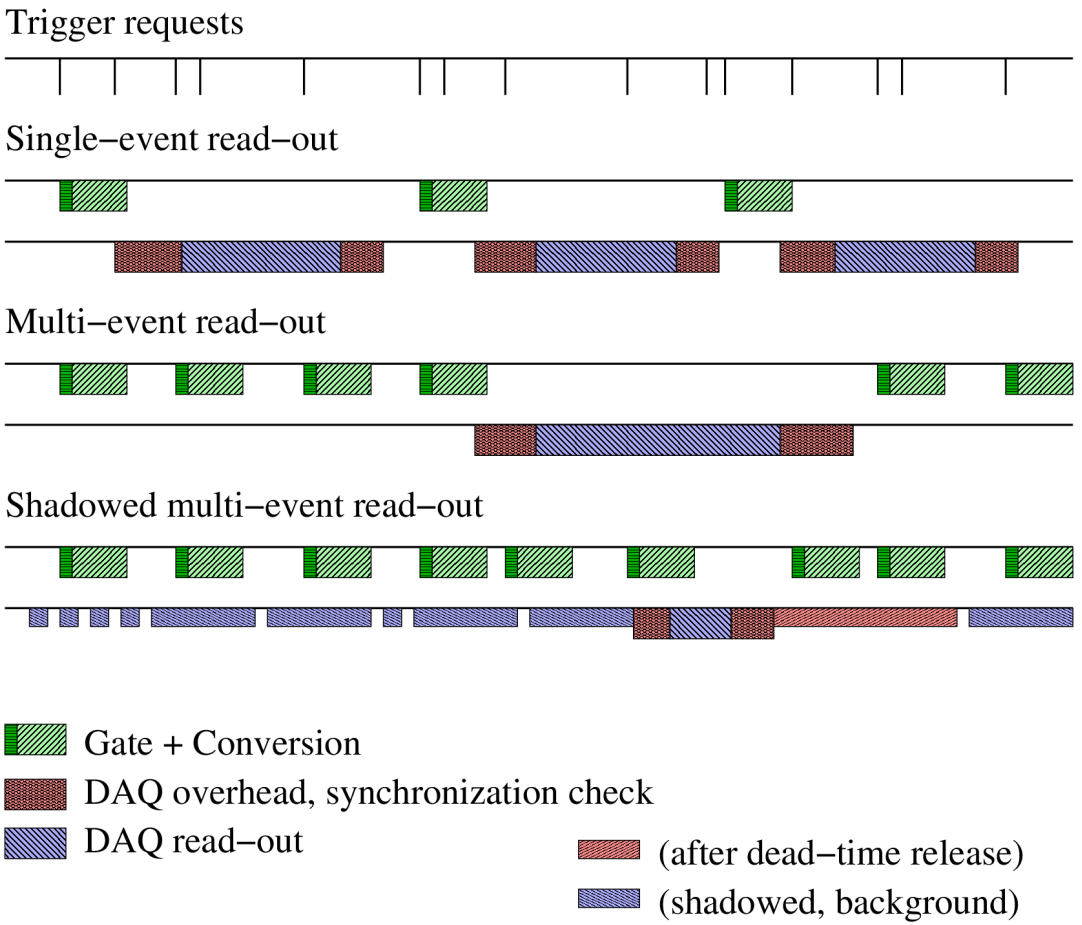
Data transfer strategies

What we want

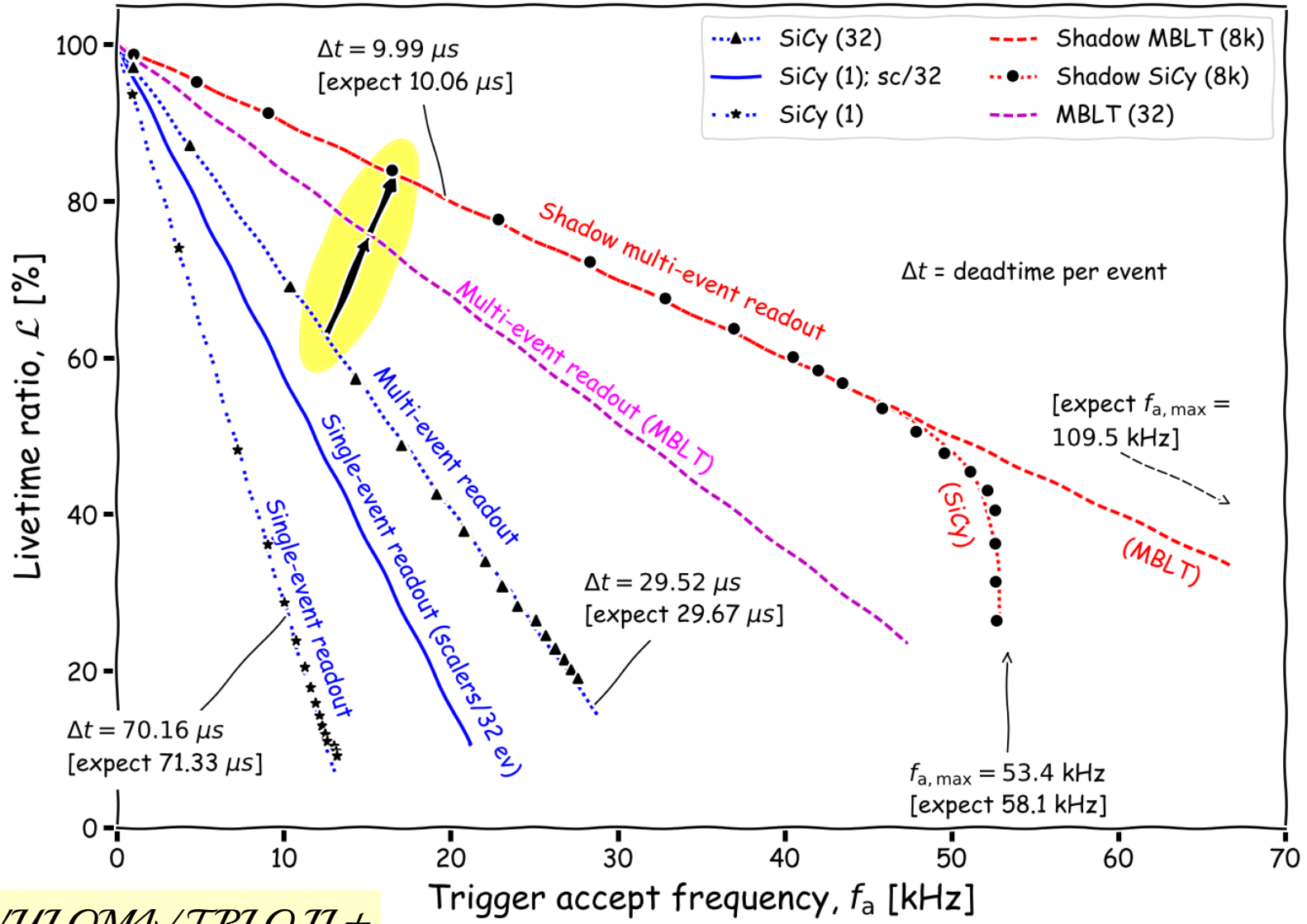
What we got

What we get

What we use

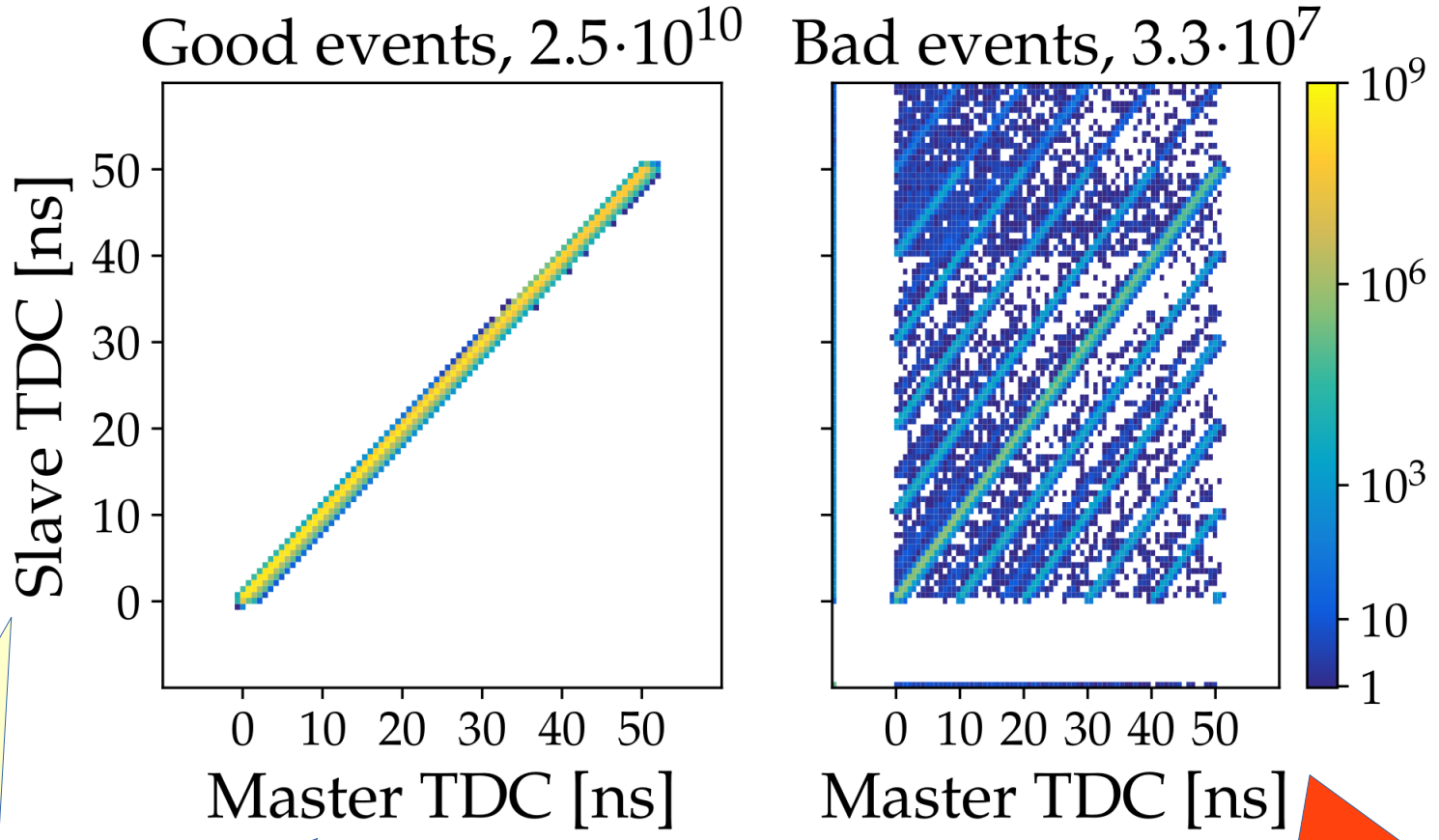


Accurate livetime predictions



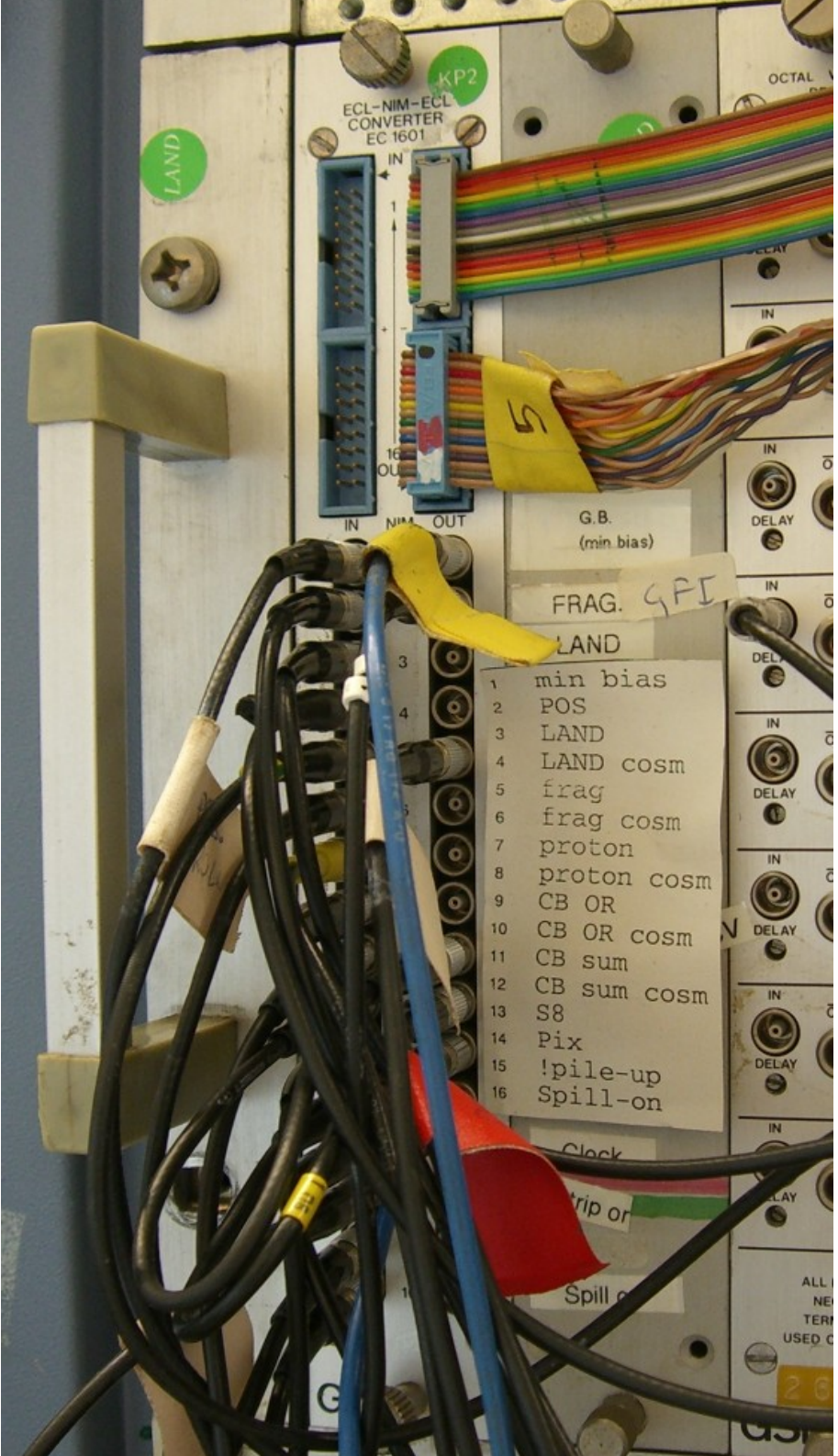
RIO4 + VULOM4/TRLO II + 6 x CAEN V785: 33 32-bit words/event

Trust, but verify



Send same signal to two systems

Intentionally injected spurious triggers, **detected** by event counter mismatches



Thank you!

Lots of FUN !

Sticky events for easier analysis

VME Readout at and Below the Conversion Time Limit

arXiv:1810.03574

Sticky subevents

- Packaged in sticky events.
- The sticky thing is the subevents.
  - Sticky = held active until replaced.
- Sticky subevents identified (as usual) by
  - `type/subtype/ctrl/crate/procid`
- Removed as active with `length = -1`.