PRS muon meeting

CERN 5 june 2001

Status of persistent digi problem

(persistent refers to both)

Stefano Lacaprara,*INFN, Padova.*

OUTLINE:

- summary of problems found so far,
- workaround and solution

Summary of problems:

- first production run with ORCA 4_4_0 on dataset mu_MB1mu_pt4 (I will refer to production which took place in Padova);
- Pythia and Cmsim done without problems;
- setup of federation, objectivity (5.2.1), and various tools;
- ~ 200 kev ooHitFormatted w/o problems;
- all event digitized: here we apply a filter to reject events w/o muon $pt > 4$GeV (non prompt muons killed by GEANT): filtering not possible during ooHit step for technical reason - ask Vincenzo to make it possible);

- apparently no problems, but I was not able to process a small fraction of digi jobs due to crashes, which I didn't investigate, just skip the runs;
- when accessing digi to perform analysis, we had crashes in specific events: we found that in these event there were "corrupted" digi (either digi with all words = "0" or strange numbers and/or string);
- mainly in Calorimetry, but also in RPC and DT (none in CSC . . . but maybe it's just statistic!);
- the same problem occurs also in dataset produces in different sites in Italy (e.g. Bologna and Roma);

- if we redo the digi on the fly (in transient mode), the digi are correct;

DT corrupted DIGI example

READING DIGI	REBUILDING DIGI
Wire Id 1 3 20 TDrift 351	Wire Id 1 3 20 TDrift 349
Wire Id 0 0 0 TDrift 0	Wire Id 1 3 21 TDrift 336
Wire Id 0 2 36 TDrift 233133	Wire Id 1 4 20 TDrift 84
Wire Id 0 0 0 TDrift 266752	Wire Id 2 1 47 TDrift 91
Wire Id 0 0 31 TDrift 1	Wire Id 2 2 47 TDrift 345
Wire Id 2 2 48 TDrift 357	Wire Id 2 2 48 TDrift 359
Wire Id 2 3 47 TDrift 112	Wire Id 2 3 47 TDrift 111

- in some sites (Torino) where a dataset of $H \rightarrow WW \rightarrow 2\mu$ were produced, they could not even write the digi for very frequent crashes;

- we try to switch to ORCA 4_5_1 and objectivity 6.0 but we have the same problem accessing the digi;
- if we read the ooHit and re-write the digi (in persistent mode) they are correct;
- investigating Torino problem we found that also the ooHit were corrupted!;
- and again, if we rewrite the ooHit starting from the fz file, the new ooHit are correct!, but when we produce a large sample, then some of the ooHits are badly written into the DB;
- the ooHit are correctly built and sent to CARF by subDetector, but sometimes are written in a corrupted way in the DB: if a ooHit is corrupted, then the digi jobs which tries to access them crashes;

- so far we found only Calorimetry bad hits, (EBRY, EFRY);

ooHit corrupted

```
[584] {
        energy = -0.0004607276059687137603 8
        time = 0.70504790544509887695
        itra = -1063174816
        mycell = {
          base = "G<\017\272"
          id = 957525139
        }
      }
      [585] {
        energy = 4.5610704421997070312
        time = 1.8834785805665887892e-05
        itra = 3
        mycell = {
          base = "EFRY"
          id = 269811737
        }
      }
```

ooHit rewritten, ok

```
[584] {
        energy = 0.0039868515567684173584
        time = 15
        itra = 6
        mycell = {
          base = "EFRY"
          id = 269811727
        }
      }
      [585] {
        energy = 0.0016076597385108470917
        time = 15
        itra = 6
        mycell = {
          base = "EFRY"
          id = 269811737
        }
      }
```

- We are not able to reproduce the problem in a controlled environment, i.e. in debugging mode, so we are not able to understand where the problem is;

Workaround and solution:

- Until we are not able to find out the origin of bad hits and digis, we can at least catch them and skip the event (they are few!): not a solution, of course, just a workaround;
- most of the bad ooHits were recognised as such during the reading phase, or, at least, something wrong with them is found, but then Calorimetry throw an unknown exception (what's the use of that!! SGRUNT), which is caught by CARF who stops the jobs in a bad way;
- I modify Calorimetry code to throw a SkipEventException, and also improve the recognising of bad ooHit;
- with this patch, I've been able to ooHitize and Digitize (no pileup) 10000 $H \rightarrow ZZ \rightarrow 4\mu$ events almost w/o crashes (only 1, an Objectivity

error), while w/o the patch I have a crash every ~ 500 events;

- I tried to access the digis, where I did expect to find something corrupted, but I could analyse all the dataset w/o problems!! WOW
- then I tried to ooHit'ze and Digitize (with full PileUp) 10000 MB $\longrightarrow 1\mu$ $pt_\mu > 1$GeV, but when accessing them I'm having crashes (and unknown exceptions re-SGRUNT);
- anyway I'm pretty confident that most (at least) of bad digis can be caught before use, and then a SkipEventException can be thrown (or we can produce a list of bad events and invalidate them, it should be possible);

Conclusion:

- All the problems we have come from corrupted objects (ooHit and Digi) written in the databases;
- in spite of big effort, we did not understand the origin of this corruption, neither we are able to reproduce it;
- the problem is not in fz file, neither in code from subdetector (e.g. digi maker), because transient object are always correct, and also persistent one can be re-written w/o problem;
- the problem is not related to geometry file (even if the one that we used (cmsim121) has a problem): e/γ people use the same one;
- it does not come from the PileUp dataset we use, because problem arise

also w/o PU;

- we must understand what's different between INFN and rest of the world, where none of these problems has been found;
- a workaround is to protect ORCA code when dealing with corrupted objects, forcing to skip the bad event: this seems promising even if not yet full available;