Trains status&tests

M. Gheata

Train types run centrally

- FILTERING
  - Default trains for p-p and Pb-Pb, data and MC (4)
    - Special configuration need to be applied for the different cases (pre-configure stage)
    - Std AOD, muons and vertexing
    - Running automatically after reco
  - On-demand filtering using special tender settings
    - Generally for pass2
    - Std AOD, muons and vertexing
  - Quite stable code, to be moved in Release
  - Vertexing is cuts dependent (needs many passes), to be done on demand as post-filtering from now on (?)
  - Memory stable (~2GB) but CPU and disk consuming
  - A queue of trains according priorities for non default trains (?)
    - Discussed and approved weekly
- PWG trains
  - Just started to be run centrally, finished tests
  - Probably many trains per PWG needed, managed by PWG

PWG1 QA train

- 20 different modules, 1 to come (FMD)
  - Heavy memory requirements (above 3GB)
    - Cleanup needed, see next slides
  - Train gets kicked out by some sites
- Merging is heavy for runs with too many chunks
  - Memory in the last merging step gets too big
  - Process only 10% of chunks
- Some new trigger classes used recently (CINT7, EMC7, …)
  - QA train was set for kMB, now we need to run it with kAnyInt plus other triggers
    - All tasks requesting a special trigger should filter from the trigger mask and fill separate histograms (within the memory limit)
    - Requirements described in Savannah bug #84007
- To be run with the Release

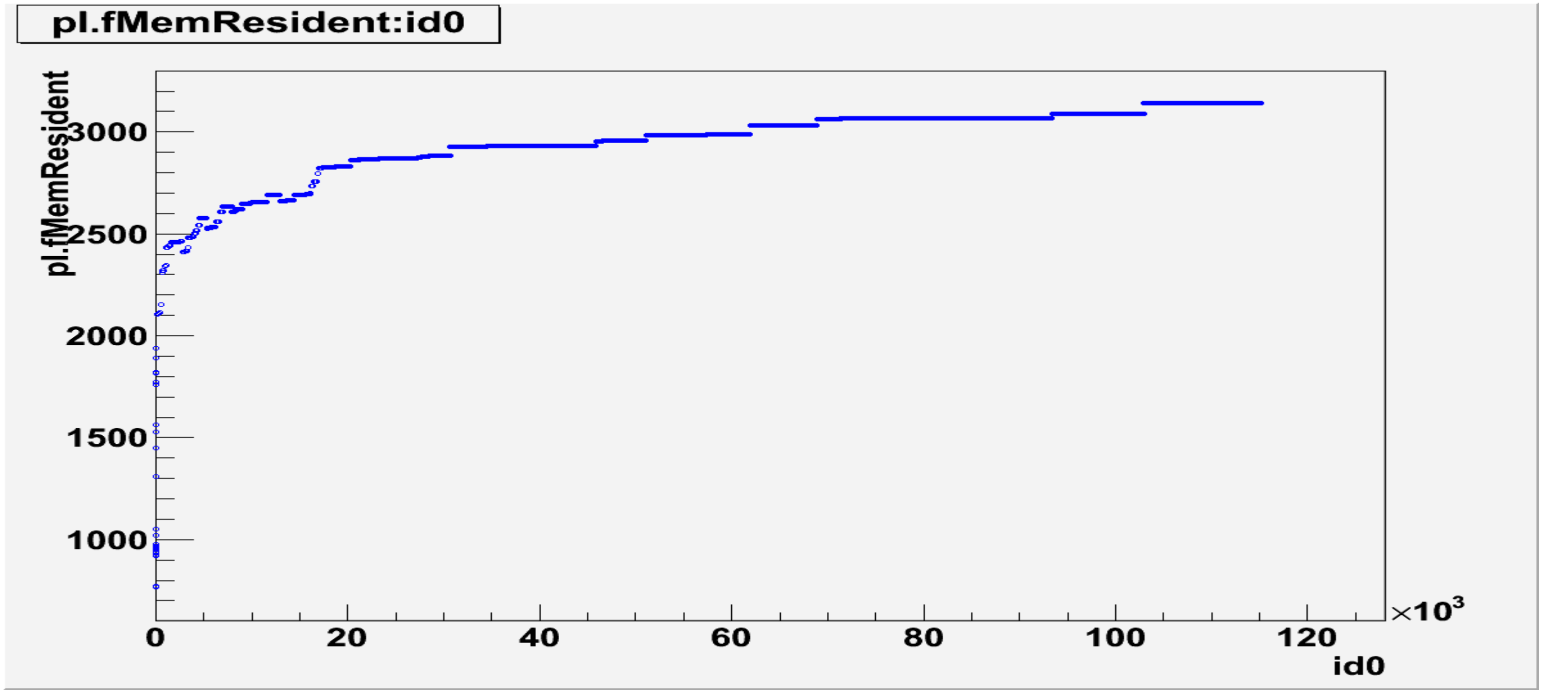
Testing QA wagons

- Prepare checking scripts to be run on a test machine
  - Testing each QA wagon independently on a local dataset
  - Checking is based on syswatch.root
  - Future support in the alien plug-in for producing the scripts
- Simple check and fit of memory profiles extracted by the analysis manager
  - Doing a linear fit in a specified range (stable regions) and extracting memory value + leak
  - Subtracting the baseline given by EventSelection task + OCDB connect
  - Not easy to set-up automatically due to memory jumps

Most recent checks

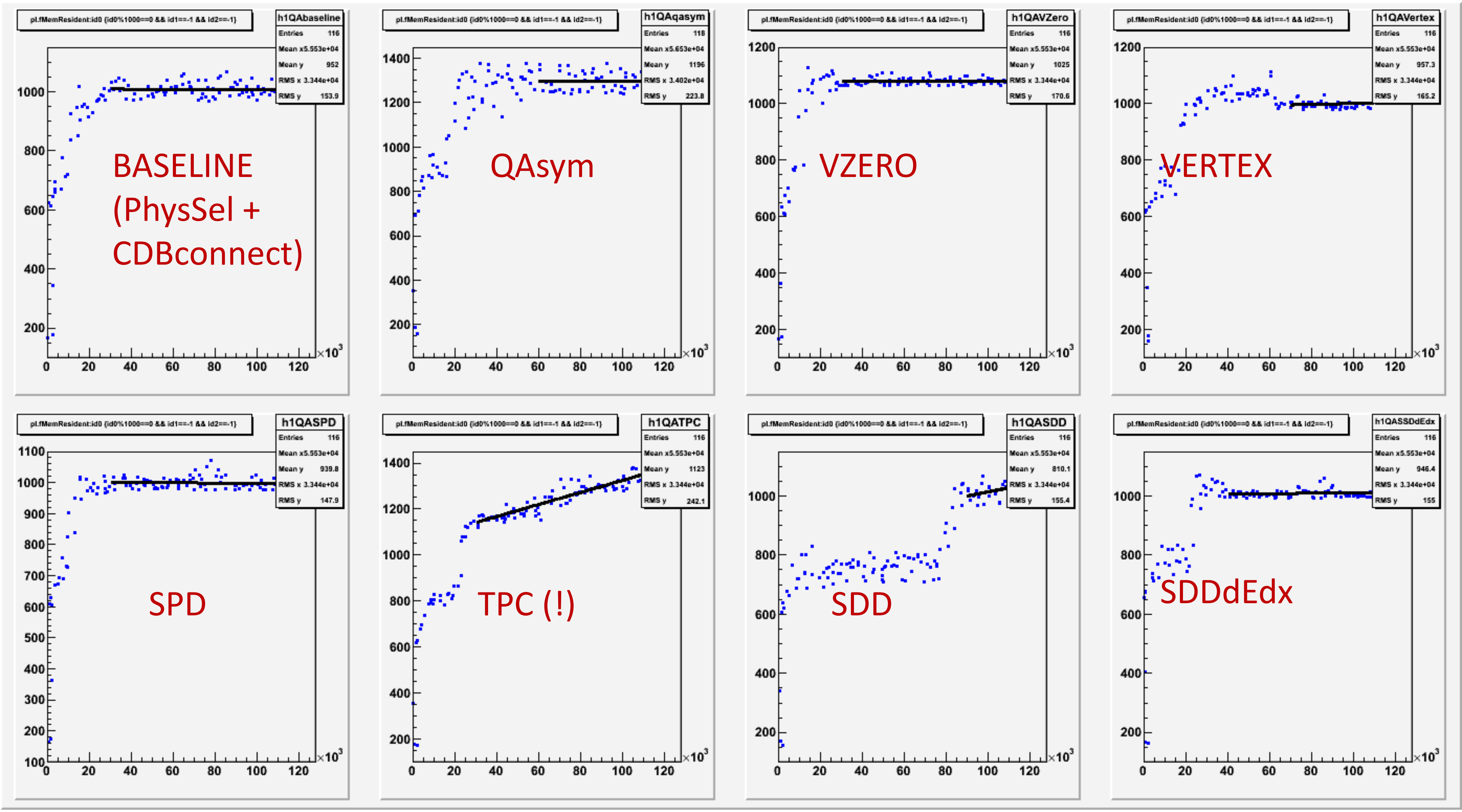
- Versions tested on pcaliense05
  - Root v5-28-00e
  - AliRoot v4-21-28-AN
- Run tested: 152371 from LHC11c
  - 120K events
- Wagons that did not finished the test:
  - ZDC (segfault) – savannah #78558
- 20 wagons checked in parallel (few hours)

Resident memory for the train

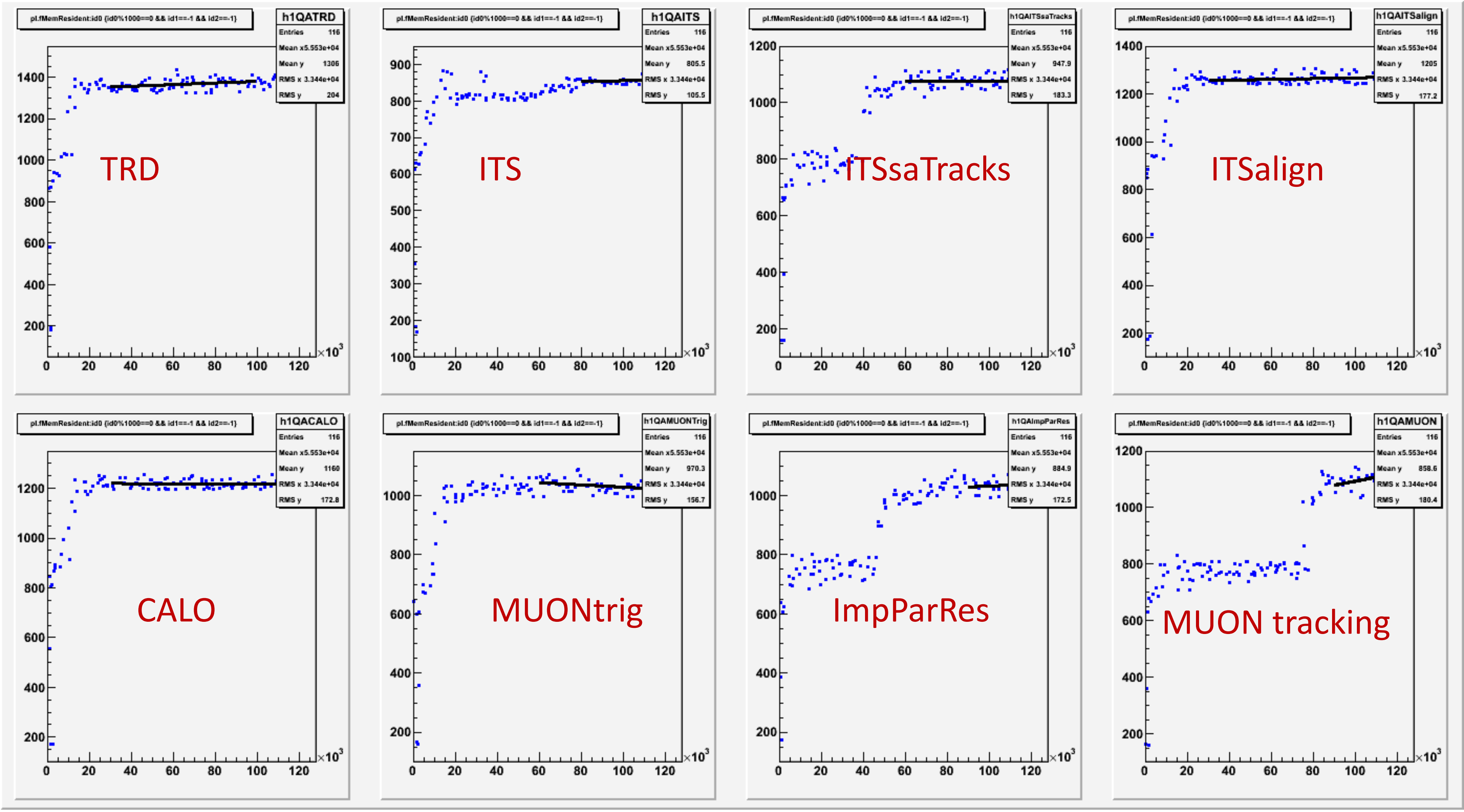


- Train run on 50 local files (~ twice the size of QA jobs)
- Some leaks make the memory go above 3 GB

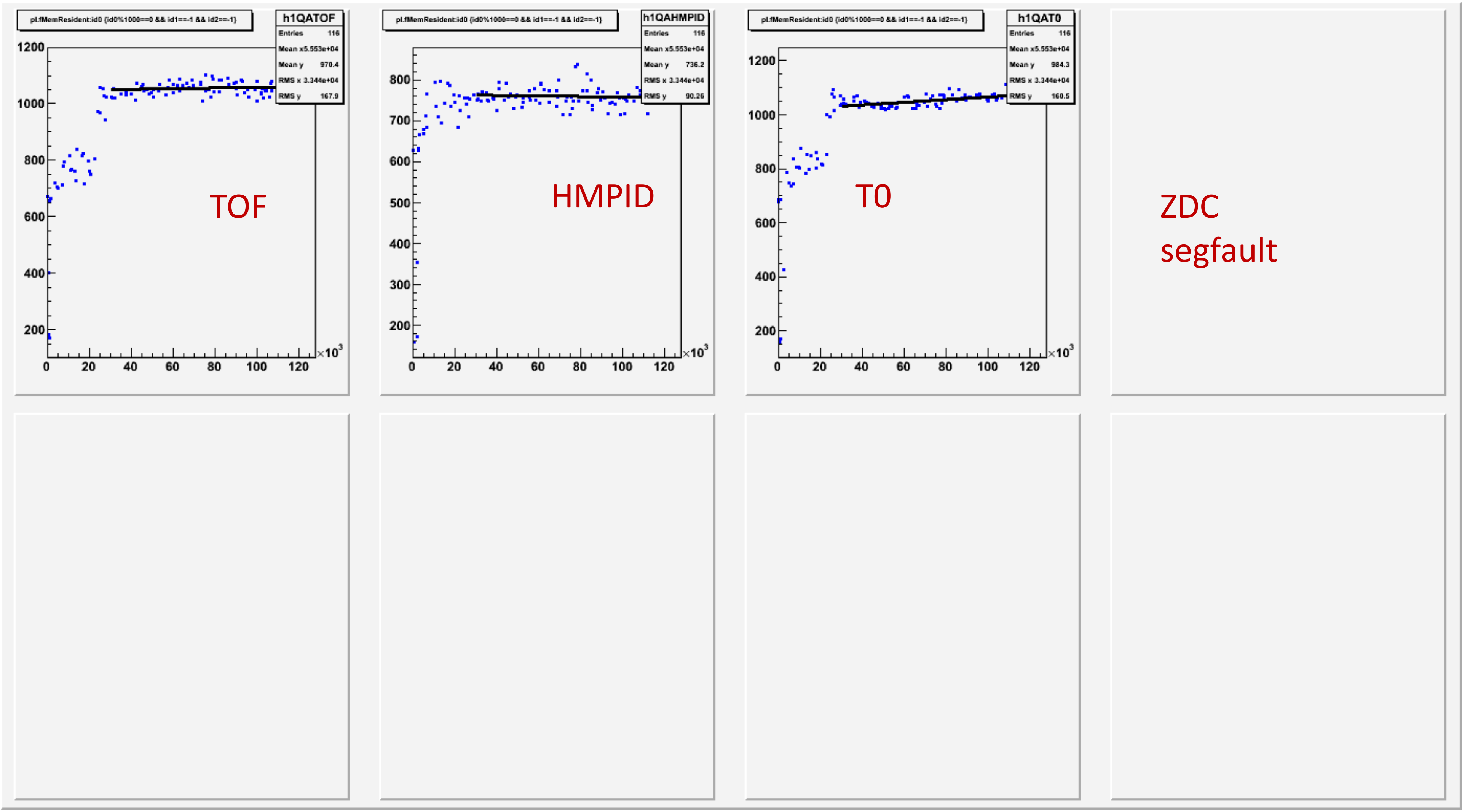
Resident memory (1)



Resident memory (2)



Resident memory(3)



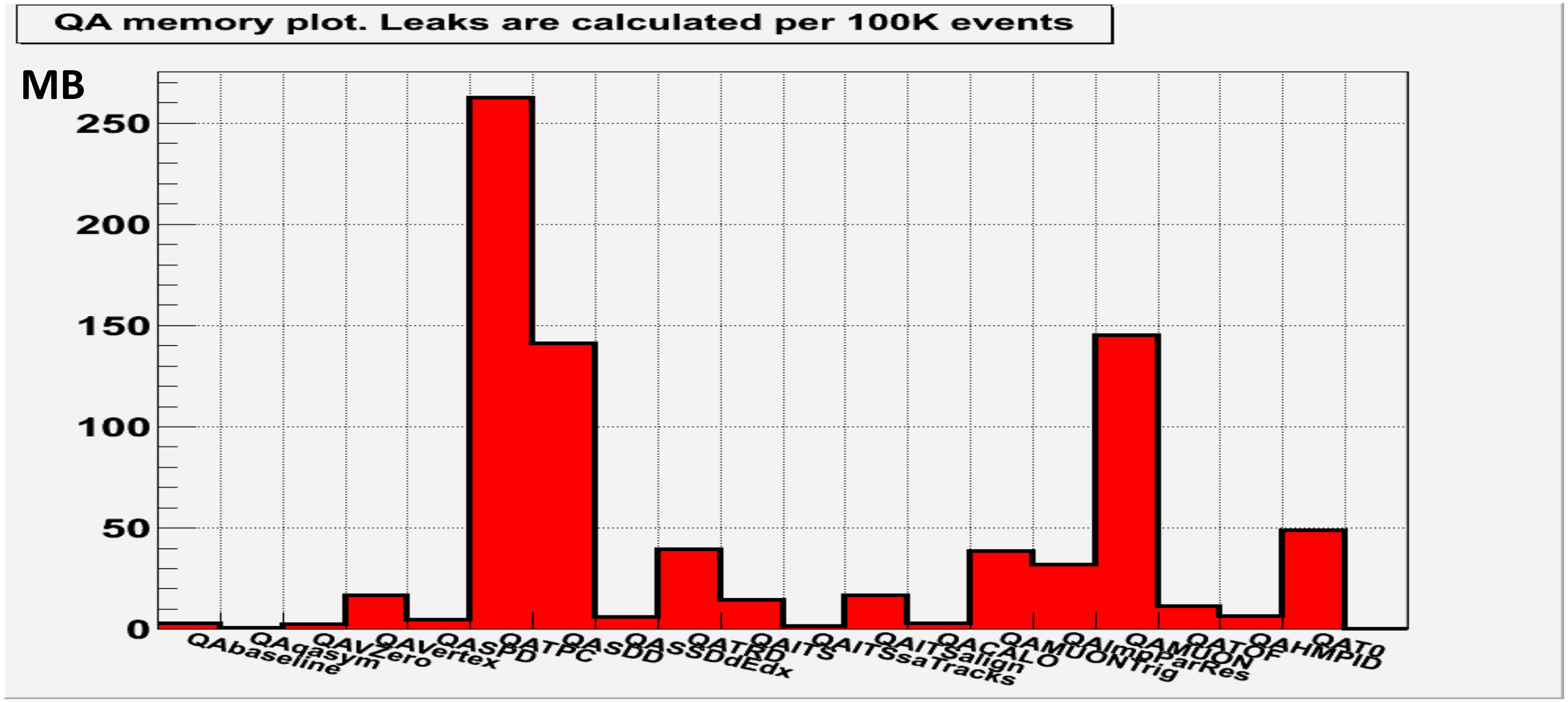
TOF

HMPID

T0

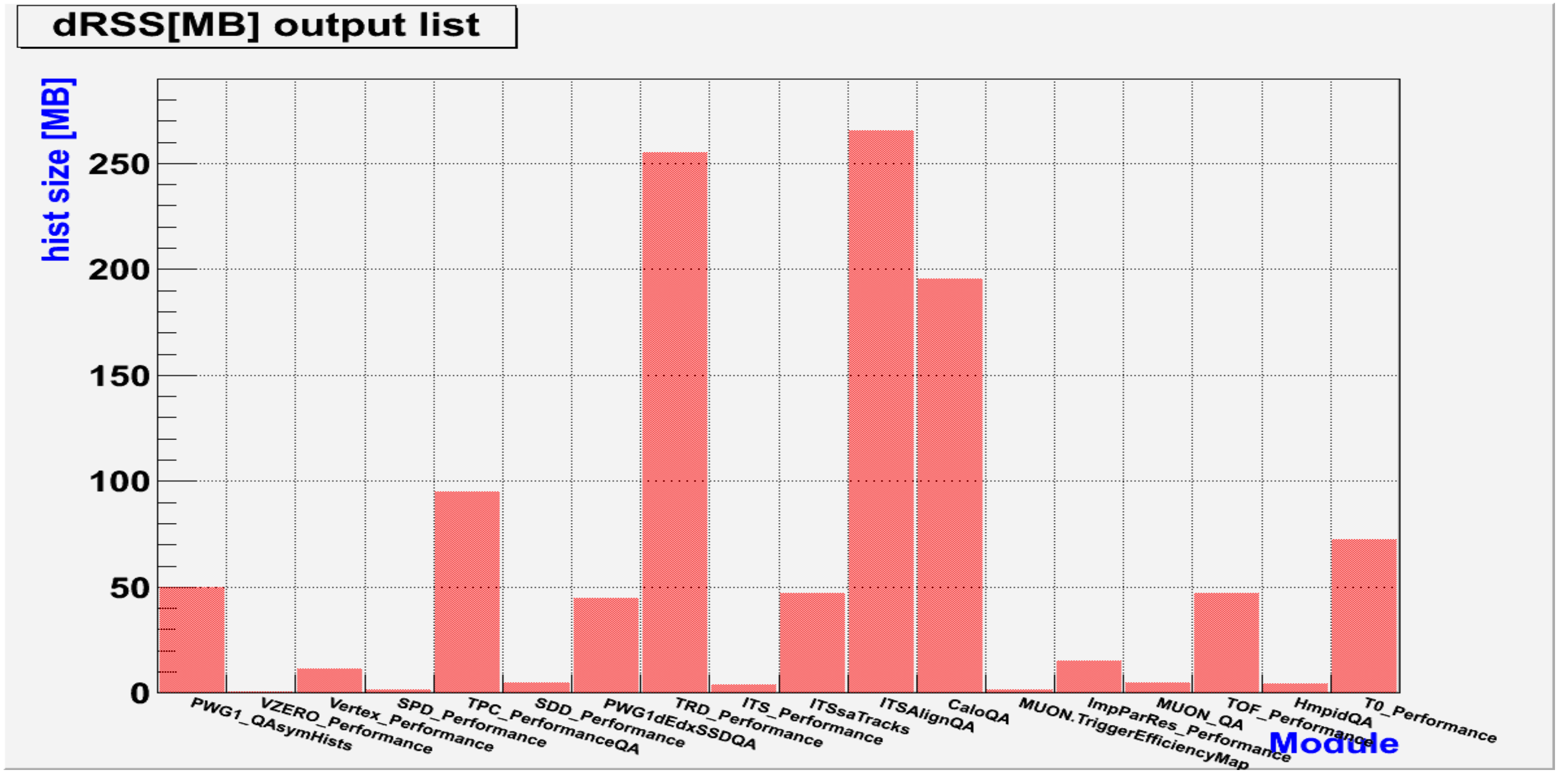
ZDC
segfault

Train leak plot



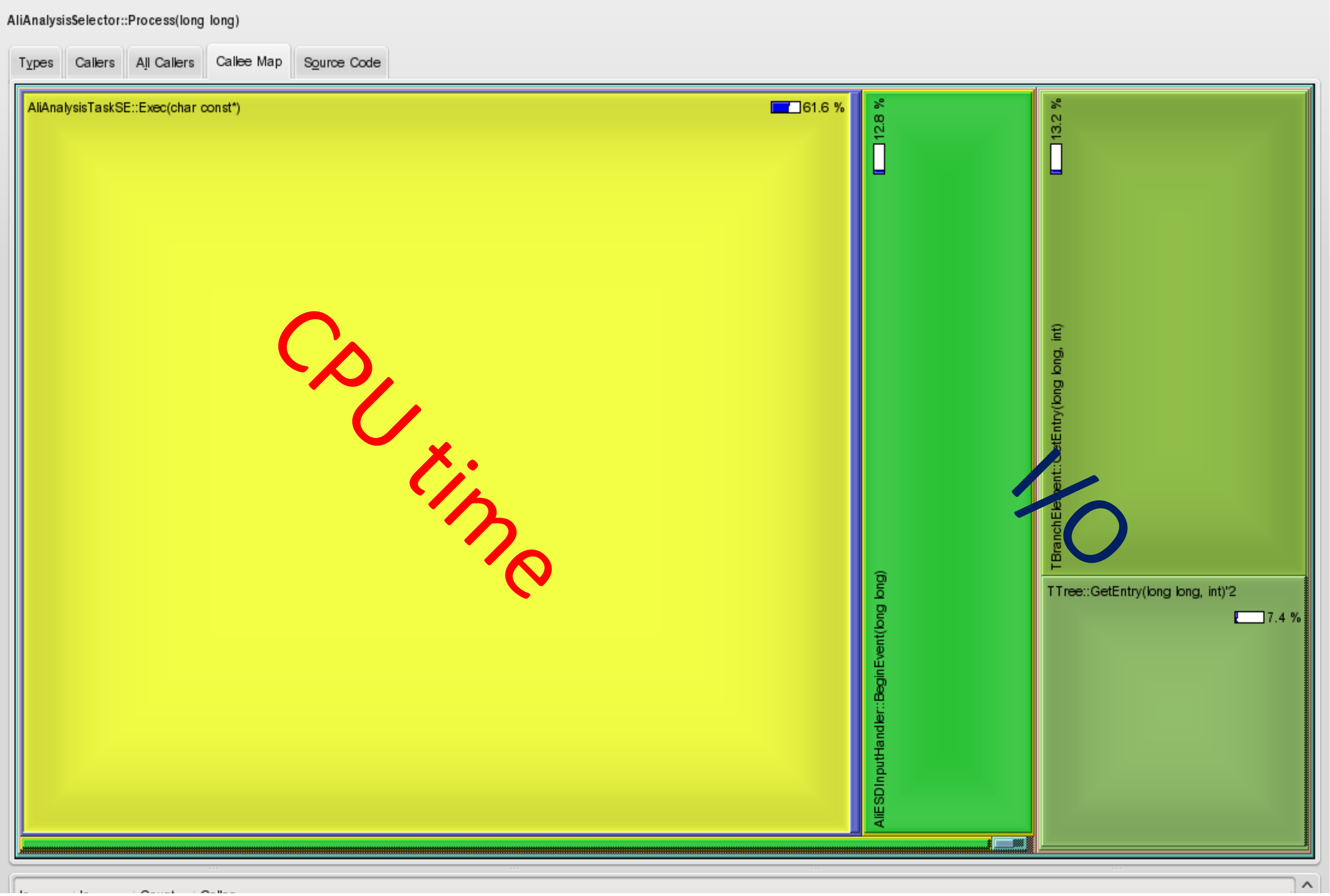
- Fit parameters (p0 + p1*event) for all memory plots
- p0 should subtract the baseline, not plotted here
- p1 (slope) plotted per 100K events (2x what the train usually processes per job)
- Some leaks visible, at a level that does not affect the job (except TPC - more than 200 MB per 100K events)

Output size in memory



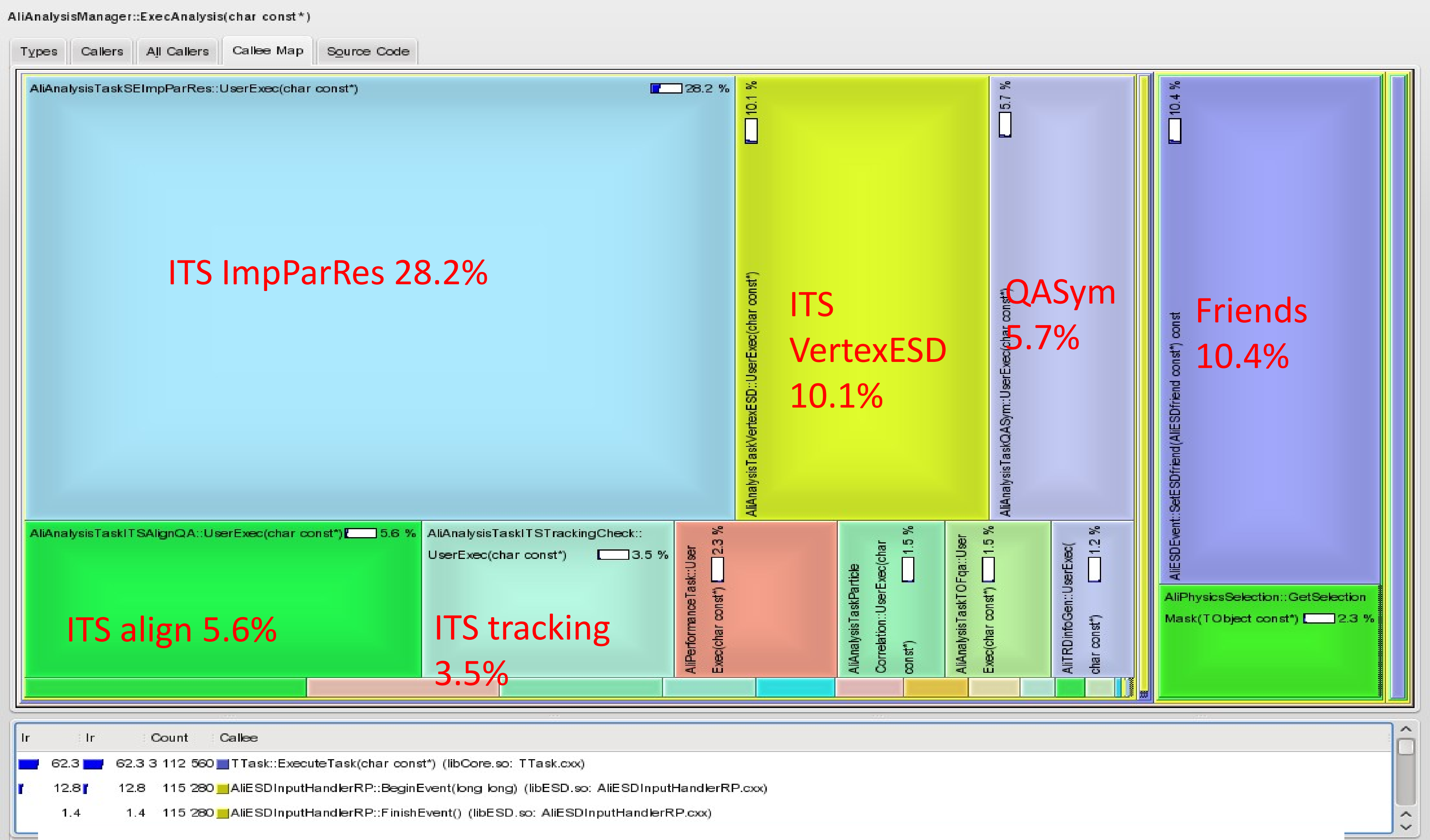
- Looping task directory in output file, reading all keys in memory
- Making the difference between the resident memory after and before reading histograms in memory
- Sum is ~1.1 GB

Performance checks


AliAnalysisSelector::Process(long long)
Types
Callers
All Callers
Callee Map
Source Code
AliAnalysisTaskSE::Exec(char const*)
61.6 %
12.8 %
AliESDInputHandler::BeginEvent(long long)
13.2 %
TBranchElement::GetEntry(long long, int)
TTree::GetEntry(long long, int)'2
7.4 %
CPU time
I/O

- Local dataset on a test machine, processing took 2 days for ~100K events
- valgrind –tool=callgrind aliroot –b –q QA.C
- 61.5 % CPU, 38.5 % I/O   (ratio is different if accessing remote files)

CPU performance per task



AliAnalysisManager::ExecAnalysis(char const*)

Types
Callers
All Callers
Callee Map
Source Code
AliAnalysisTaskSEImpParRes::UserExec(char const*) 28.2 %
AliAnalysisTaskVertexESD::UserExec(char const*) 10.1 %
AliAnalysisTaskQASym::UserExec(char const*) 5.7 %
AliESDEvent::SetESDfriend(AliESDfriend const*) const 10.4 %
AliAnalysisTaskITSAlignQA::UserExec(char const*) 5.6 %
AliAnalysisTaskITSTrackingCheck::UserExec(char const*) 3.5 %
AliPerformanceTask::UserExec(char const*) 2.3 %
AliAnalysisTaskParticleCorrelation::UserExec(char const*) 1.5 %
AliAnalysisTaskTOFqa::UserExec(char const*) 1.5 %
AliTRDinfoGen::UserExec(char const*) 1.2 %
AliPhysicsSelection::GetSelectionMask(TObject const*) 2.3 %

ITS ImpParRes 28.2%

ITS VertexESD 10.1%

QASym 5.7%

Friends 10.4%

ITS align 5.6%

ITS tracking 3.5%

Ir	Ir	Count	Callee
62.3	62.3	3 112 560	TTask::ExecuteTask(char const*) (libCore.so: TTask.cxx)
12.8	12.8	115 280	AliESDInputHandlerRP::BeginEvent(long long) (libESD.so: AliESDInputHandlerRP.cxx)
1.4	1.4	115 280	AliESDInputHandlerRP::FinishEvent() (libESD.so: AliESDInputHandlerRP.cxx)

- ITS tasks use ~50% of the CPU time

Trains management

- Trains can be now prepared and checked using the alien plug-in
- Submission and management now done mostly by Costin and Latchezar
  - We need a simple system that can be used by train administrators
    - Start, stop, cleanup, input datasets
  - Defining a queue of central non-default trains

Conclusions

- Currently implemented checking macros for QA train
- Many central trains and more to come, need some automated assembly and checking procedure
- Make train administration easier and allow more people using it