Performance monitoring of the software frameworks for the LHC experiments

Diana-Andreea Popescu

CERN openlab
27th of August 2010

oTN-2010-01 openlab Summer Student Report

Performance monitoring of the software frameworks for the LHC experiments

Student:Diana-Andreea Popescu
Supervisor:Andrzej Nowak

27 August 2010
Version 1
Distribution: **Public**

Abstract

This document presents the extraction, maintenance and performance monitoring of the latest benchmarks representative of the 4 major software experiment frameworks at CERN, namely ATLAS, ALICE, CMS and LHCb. The benchmarks contain all event processing steps. The performance monitoring tasks for the software experiments were carried using the *Intel Performance Tuning Utility* and *perfmon*.

Introduction

The first chapter offers an overview of the typical execution stages in the software frameworks for the experiments. The second chapter focuses on explaining the necessary steps for setting up the frameworks and on describing how to run the frameworks. The third chapter presents the technical approach used for monitoring the performance of the frameworks and the results that were obtained.

1 Execution stages in the software frameworks for the experiments

The model for the software frameworks is composed of execution stages, each one depending on the output generated by the previous one. These stages are:

1. Generation: is the event generation.
2. Simulation: is the transport of particles through the detector. It consists in the tracking of the particles in the detector and simulating the physics processes occurring in the experimental setup. Simulation takes the longest time in comparison to the other stages.
3. Digitization: the detector response is applied to hits previously generated in the simulation. The output is digitized data that imitates the real data coming from the real detector.
4. Reconstruction: the digits produced in the third phase are processed in order to obtain tracks and energy deposits.

Generation → Simulation → Digitization → Reconstruction

2 Software frameworks for the experiments

2.1 ATLAS

2.1.1 Introduction

ATLAS (A Toroidal LHC ApparatuS) is one of two General Purpose Detectors at the CERN Large Hadron Collider (LHC). The ATLAS computing model is based on two major blocks: the Athena software framework, with its associated modular structure of the event data model, including the software for event simulation, event trigger, event reconstruction, physics analysis tools, and the Distributed Computing tools which are built on top of Grid middleware.

Athena is a control framework and is a concrete implementation of an underlying architecture called Gaudi. The Gaudi project was originally developed by LHCb. At the present time, the Gaudi project is a kernel of software common to both experiments and co-developed, while Athena is the sum of this kernel plus ATLAS-specific enhancements.

The “Full Chain” consists of several steps: Generation, Simulation, Digitization, Reconstruction, Creating AOD and Analysis, as shown in the diagram. Each of these stages is described separately, but they can be combined if required.

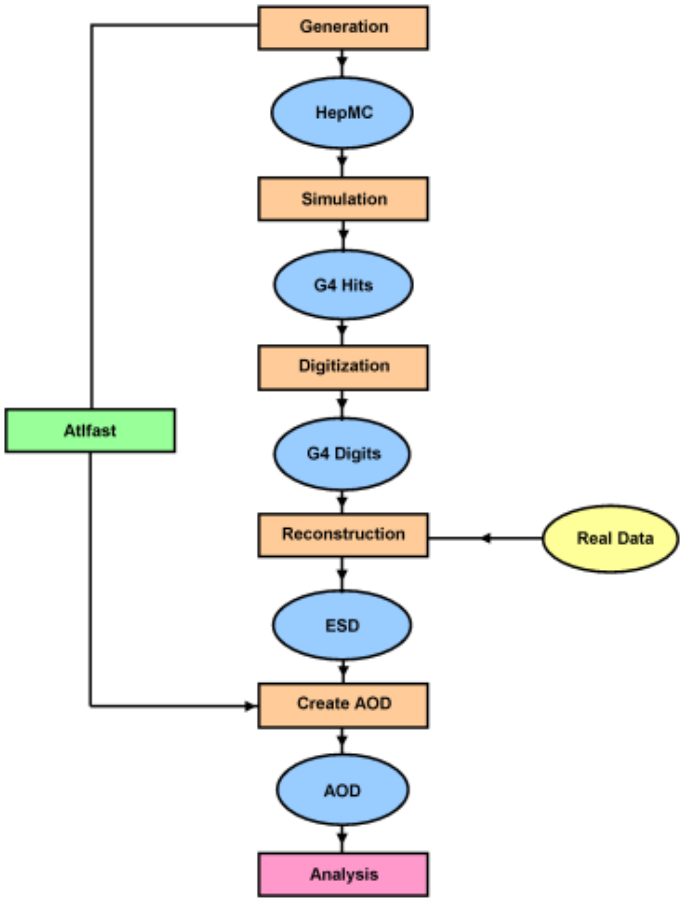

Figure 1 - Schematic representation of the Full Chain

The Event Data Model defines a number of different data formats:

* RAW:
  - "ByteStream" format, ~1.6 MB/event
* ESD (Event Summary Data):
  - Full output of reconstruction in object (POOL/ROOT) format:
    + Tracks (and their hits), Calo Clusters, Calo Cells, combined reconstruction objects etc.
  - Nominal size 1 MB/event initially, to decrease as the understanding of the detector improves
    + Compromise "being able to do everything on the ESD" and "not storage for oversize events"
* AOD (Analysis Object Data):
  - Summary of event reconstruction with "physics" (POOL/ROOT) objects:
    + electrons, muons, jets, etc.
  - Nominal size 100 kB/event (currently roughly double that)
* DPD (Derived Physics Data):
  - Skimmed/slimmed/thinned events + other useful "user" data derived from AODs and conditions data
    + DPerfD is mainly skimmed ESD
  - Nominally 10 kB/event on average
    + Large variations depending on physics channels
* TAG:

- Database (or ROOT files) used to quickly select events in AOD and/or ESD files.

All the execution stages - event generation, detector simulation, digitization and reconstruction - are performed in Athena. Athena can be run as follows: `athena.py job_option_file`. For running the jobs separately, Job Transformations can be used. These are Python scripts that automatically configure and run Athena jobs.

For running Atlas and for checking out packages from the ATLAS repository, the user must have an account in the group ZP.

Information on how to set up ATLAS and how to run it can also be found in the Appendix.

2.1.2 Event generation

The generation is run using the Evgen_trf.py job transform, which is available from the command line after a release has been set up. The command can be run as follows:

Evgen_trf.py [options] <ecmenergy> <runnumber> <firstevent> [maxevents] <randomseed> <jobconfig> <outputevgenfile> [histogramfile] [ntuplefile] [inputgeneratorfile] [evgenjobopts]

```
Evgen_trf.py --omitvalidation=testEventMinMax ecmEnergy=7000. runNumber=105144
firstEvent=1 maxEvents=10 randomSeed=1324354657
jobConfig=MC09JobOptions/MC9.105144.PythiaZee.py outputEvgenFile=pythia.pool.root
```

The arguments are:

- the center of mass energy in GeV, set to 7000;
- the run number of the process to be simulated (each run number corresponds to one physics process);
- the number of the first event;
- the number of events to be written in the output file;
- a ten-digit random number which is passed to the event generator(s) - it should begin with 1-;
- the name of the job options file containing the physics settings which are passed to the generators;
- the name of the output file containing the generated events.

In order to modify the number of generated events, it is necessary to download the job option file: `get_files -jo FileName` and add at the end of the file the following line: `evgenConfig.minevents = number of events.`

Default, the number of generated events is 5000. For seeing the complete list of processes, the user can list the following directory:
`/afs/cern.ch/atlas/software/releases/VersionNumber/AtlasOffline/VersionNumber/InstallArea/jobOptions`

To find out what is the meaning of the options, the following command can be used: `Evgen_trf.py -h.` The output of the command can be found in the Appendix.

2.1.3 Simulation

The Geant4 simulation is run using the csc_atlasG4_trf.py job transform, which is available from the command line after a release has been set up. To find out what the options mean the following can be used: `csc_atlasG4_trf.py -h`.

The command can be run as follows:

```
csc_atlasG4_trf.py inputEvgenFile=pythia.pool.root outputHitsFile=g4hits.pool.root
maxEvents=10 skipEvents=0 randomSeed=1234 geometryVersion=ATLAS-GEO-06-00-00
conditionsTag=OFLCOND-SIM-BS7T-02 physicsList=QGSP_BERT jobConfig=NONE
```

The arguments are:

- the evgen file produced in the generation stage;
- the name of the output file;
- the number of events to process;
- the number of events at the beginning of the evgen file to skip;
- a random number seed offset to use during simulation;
- the detector layout to use for simulation;
- the physics list to use (should almost always be 'QGSP_BERT');
- any additional job configuration fragments

2.1.4 Digitization

The digitization is run using the csc_digi_trf.py job transform which is available from the command line after a release has been set up. The command can be run as follows:

```
csc_digi_trf.py inputHitsFile=g4hits.pool.root outputRDOFile=g4digi.pool.root
maxEvents=-1 skipEvents=0 geometryVersion=ATLAS-GEO-06-00-00  conditionsTag=OFLCOND-DR-
BS7T-ANom-06 digiseedoffset1=11 digiseedoffset2=22
```

The arguments are:

- the Geant4 Hits file produced in the simulation stage;
- the name of the output RDO file;
- the number of events to process;
- the number of events at the beginning of the hits file to skip;
- the detector layout to use for simulation;
- two random-seed offsets (which should be changed each time you run the transform);
- any additional job configuration fragments.

This will digitize all the Geant4 events in the input file and run the LVL1 Trigger simulation using the default Trigger Menu for the current release. For not running LVL1 Trigger simulation, the following command line option should be added:

TriggerConfig=NONE

In this case the LVL1 inputs will be simulated, but not the LVL1 Trigger itself.

To find out what the options mean, the following can be used: `csc_digi_trf.py  -h.`

2.1.5 Reconstruction

The reconstruction is run using the Reco_trf.py job transform which is available from the command line after a release has been set up. The command can be run as follows:

```
Reco_trf.py  inputRDOFile=g4digi.pool.root outputESDFile=esd.pool.root maxEvents=-1
skipEvents=0 geometryVersion=ATLAS-GEO-06-00-00 conditionsTag=OFLCOND-DR-BS7T-ANom-06
```

The arguments are:

- the input file produced in the digitization stage;
- the name of the output ESD file;
- the number of events to process;
- the number of events at the beginning of the hits file to skip;
- the detector layout to use for simulation;
- the COOL conditions database tags to use.

An Analysis Object Data (AOD) can be produced at the same time as producing ESD by specifying the following command line option `outputAODFile=aod.pool.root`. AOD is a summary of the reconstructed ESD.

It is often desirable to run jobs with the same configuration as for given official ATLAS productions. The parameters of the production tasks are stored in the AMI database. The following commands work with a recent release:

```
(1) Reco_trf.py AMI=q108
(2) Reco_trf.py inputBSFile=aFile.data AMI=q108
(3) Reco_trf.py AMI=q108 outputESDFile=myESD.pool.root
(4) Reco_trf.py AMI=q108 maxEvents=10
(5) Reco_trf.py AMI=q108 append_preExec=rec.doDetailedPerfMon=True
```

For command (1), the full configuration is taken from AMI, including the input/output file types which take default values. For (2) and (3), Reco_trf will use the input and output specified by the command and ignore those of AMI. For (4) and (5), the physics configuration from AMI is modified by additional arguments from the command.

The rules are the following:

- if user command specifies an input, it will be used. Otherwise, the AMI type with a default value will be used.
- if user command specifies any output(s), it will be used and all AMI outputs will be ignored. Otherwise, all output types provided by AMI with default values will be used.
- the physics configuration is taken from both AMI and the command. In case of conflict, the command argument overwrites the one of AMI (4), unless the append option is used (5).

2.2 ALICE

2.2. 1 Introduction

For the ALICE (A Large Ion Collider Experiment) experiment, the LHC will collide lead ions to recreate the conditions just after the Big Bang under laboratory conditions. The data obtained will allow physicists to study a state of matter known as quark‑gluon plasma, which is believed to have existed soon after the Big Bang.

2.2.2 ALICE software

The ALICE software framework has only two execution stages: Simulation and Reconstruction.The simulation framework covers the simulation of collisions and generation of the emerging particles, the transport of particles through the detector, the simulation of energy depositions (hits) in the detector components, their response in form of summable digits, the generation of digits from summable digits with the optional merging of underlying events and the creation of raw data. The inputs to the reconstruction framework are digits in root tree format or raw data format. First, a local reconstruction of clusters is

performed in each detector. Then vertexes and tracks are reconstructed and particles types are identified. The output of the reconstruction is the Event Summary Data (ESD).

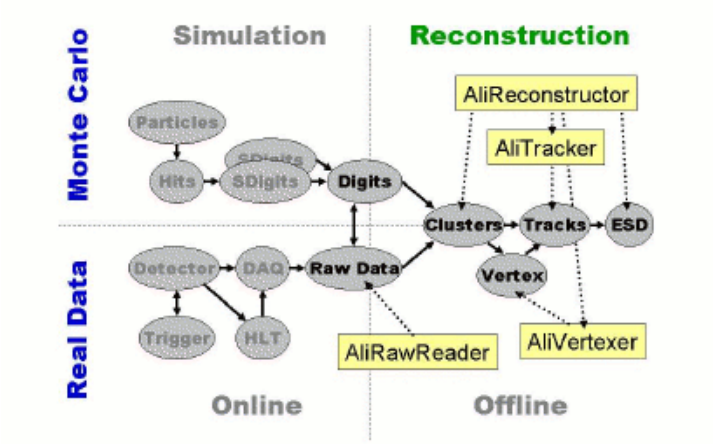

Figure 2 - Schematic representation of the execution stages for ALICE

2.2.3 Installing and running ALICE

The installation procedure for Alice is very straightforward and can be seen in the Appendix. There was an issue with a missing symlink in /usr/lib64: `/usr/lib64/libg2c.so -> /usr/lib64/libg2c.so.0`, but is was solved by creating the necessary simlink.

For running Alice, use the script `alice_run.sh`. Syntax:

```
alice_run.sh -n number_of_events [[-p] || [-c pfmon_command]] [-o output_path]
```

-n number_of_events: The number of events that will be generated.
-p: Activate profiling with pfmon (default flags).
-c pfmon_command: Activate profiling with pfmon specifying the pfmon command (use " ").
-o output_path: The path where you want to generate the output files.

This will run both the simulation and reconstruction phases. The simulation part is run using `aliroot -b -q mySim.C` and the reconstruction part is run using `aliroot -b -q myRec.C.`

The scripts for running and installing ALICE are written by Jose Dana.

2.3 CMS

2.3.1 Introduction

The CMS experiment uses a general-purpose detector to investigate a wide range of physics, including the search for the Higgs boson, extra dimensions, and particles that could make up dark matter. Although it has the same scientific goals as the ATLAS experiment, it uses different technical solutions and design of its detector magnet system to achieve these.

2.3.2 CMS software

The procedure for installing CMS is described in the Appendix. The instructions are for installing and running tests on a standalone machine, without a network connection to the external world.

The command `cmsDriver.py` generates python configuration files. To find out what the options mean, the following can be used: `cmsDriver.py -h`. Usually, CMS software is run in 2 steps:

- step 1 - event generation, simulation, digitization and L1/HLT trigger
- step 2 - event reconstruction (runs on files output of "step 1")

Step 1:

```
cmsDriver.py MinBias.cfi -s GEN,SIM,DIGI,L1,DIGI2RAW,HLT:1E31,RAW2DIGI,L1Reco -n 500 --eventcontent FEVTSIM --conditions auto:mc --relval 10000,100 --datatier 'GEN-SIM-DIGI-RAW-HLT' --no_exec
```

The arguments are:

- the steps that will be run
- the number of events
- what event content to write out
- what conditions to use
- set total number of events and events per job
- what datatier to use
- do not exec cmsRun, just prepare the python configuration file

Step 2:

```
cmsDriver.py recominbias -s RAW2DIGI,RECO -n -1 --filein file:500evt_MinBias_cfi_GEN_SIM_DIGI_L1_DIGI2RAW_HLT_RAW2DIGI_L1Reco.root --eventcontent RECOSIM --conditions auto:mc --no_exec
```

The arguments are:

- the steps that will be run
- the number of events
- the name of the input file
- what conditions to use
- do not exec cmsRun, just prepare the python configuration file

For running the jobs, the command `cmsRun` is used.

```
cmsRun FileName.py
cmsRun MinBias_cfi_GEN_SIM_DIGI_L1_DIGI2RAW_HLT_RAW2DIGI_L1Reco.py
```

The "step 2" reconstruction jobs run much faster than the "step 1" jobs, which are dominated by the Geant4 simulation.

2.4 LHCb

2.4.1 Introduction

The LHCb (Large Hadron Collider beauty experiment) is specialized for precise measurements of CP violation and rare decays.

2.4.2 Installing LHCb

The LHCb applications use CMT for code management and are therefore easy to set up and configure. Information on how to set up LHCb applications and how to run them can also be found in the Appendix.

The information that has to be passed to `SetupProject` is the name of the application (or project) the user wants to deploy. The version can be specified as well, but if it is not specified, `SetupProject` will use the latest available. The list of available versions can be obtained with the command line option `--list-versions`, that can be abbreviated with `--list`.
If the user wants to modify options or to create new packages, he will need to create a user local project. In order to do so, `SetupProject` has to be called with the option `--build-env` and it will create the project directory. The name of the project and the version that will be used have to be specified. If the version is omitted, `SetupProject` will ask which one to use. The location in which the user project is created is defined by the variable `User_release_area.`

For checking out packages, the following command can be used: `getpack PackageName`.

For running LHCb applications, the command `gaudirun.py` is used. The arguments are option files, as described below.

2.4.3 Generation and Simulation - Gauss

Gauss is the LHCb Simulation software, built on the Gaudi Framework. It consists of two phases: a first phase where the events are generated (e.g. pp collisions at 14 TeV) and a second phase where the particles are propagated through the LHCb detectors.

For modifying the number of events, the file `Gauss-Job.py` should be edited by changing the following line:

```
nEvts = number of events
```

Example: `gaudirun.py $GAUSSOPTS/Gauss-MC09.py $DECFILESROOT/options/11144101.opts $GAUSSOPTS/Gauss-Job.py`

The arguments are:

- a standard Gauss jobs with the MC09 geometry
- the chosen event type
- what is specific to the job (i.e. number of events, name of output file,...)

Options for different configuration of the application are provided. The configuration for different event types is provided in the DecFiles configuration package.

2.4.4 Digitization - Boole

Boole is the LHCb digitization software, built on the Gaudi framework. The Boole digitization is the final stage of the LHCb detector simulation.

In order to setup Boole to digitize the events generated earlier, the `MC09-Files.py` file should be edited by changing the input data file to the FileName.sim file generated earlier in Gauss.

```
datasetname = 'FileName'
```

```
EventSelector().Input = ["DATAFILE='PFN:$HOME/cmtuser/Gauss_vNrP/Sim/Gauss/options/" +
datasetName + ".sim' TYP='POOL_ROOTTREE' OPT='READ'
```

Boole can be run with `gaudirun.py $BOOLEOPTS/Boole-MC09-WithTruth.py $BOOLEOPTS/MC09-Files.py`

2.4.5 Reconstruction - Brunel

Brunel is the LHCb event reconstruction application, based on the Gaudi and LHCb frameworks. Brunel can process either real data from the LHCb DAQ system, or the output of the detector digitization application (Boole).

In the options directory, the input file should be set to the output produced by Boole, i.e.FileName.digi, in MC09-Files.py. The file name to change is the last one which is by default "30000000-100ev-20090407-MC09" and also the EventSelector line underneath.

Brunel can be run with `gaudirun.py $BRUNELOPTS/Brunel-MC09-WithTruth.py $BRUNELOPTS/MC09-Files.py`

3 Performance monitoring

3.1 Performance monitoring tools

The two performance monitoring tools used in this study are the Intel *Performance Tuning Utility* and *pfmon*. Intel Performance Tuning Utility is a cross-platform performance analysis tool. The Intel Performance Tuning Utility enables collecting performance data in the following modes: sampling, statistical call graph, exact call graph, heap profiling and data access profiling.
Perfmon2 is a performance monitoring subsystem for Linux, which gives the user access to both low-level counters within the processor and a high-level analysis facility. Pfmon is the performance monitoring client working with perfmon2. It can be used to monitor unmodified binaries in its per-thread mode and it can also be used to run system wide monitoring sessions. Pfmon can monitor activities happening at the user and/or kernel level for both type of sessions. It can be used to collect basic event counts and to sample program or system execution.

3.2 Performance monitoring with SEP

The sampling collector in the Intel Performance Tuning Utility is based on SEP. The sampling mode has been used. For monitoring the frameworks, the following command line was used:

```
/opt/intel/ptu4b/bin/sep -start -ec CERN_profile -em  dts=100   -out file_name -app
application_name -args "list of application arguments"
```

- -ec → event configuration
- -em [dts=<time in milliseconds>] → enables event multiplexing
- CERN_profile – constructed by David Levinthal from Intel
- -out → specify the name of the output file where the data is collected
- -app → application name
- -args → list of application arguments

The CERN profile is listed in the appendix.

Example for ATLAS:

```
   /opt/intel/ptu4b/bin/sep -start -ec
ARITH.CYCLES_DIV_BUSY:sa=2000000,BR_INST_EXEC.DIRECT_NEAR_CALL:sa=20000,BR_INST_EXEC.IND
IRECT_NEAR_CALL:sa=20000,BR_INST_EXEC.INDIRECT_NON_CALL:sa=20000,BR_INST_EXEC.NON_CALLS:
sa=200000,BR_INST_RETIRED.ALL_BRANCHES:sa=200000,BR_INST_RETIRED.NEAR_CALL:sa=4000,BR_MI
SP_EXEC.ANY:sa=20000,CPU_CLK_UNHALTED.THREAD:sa=2000000,DTLB_MISSES.ANY:sa=200000,INST_D
ECODED.DEC0:sa=2000000,INST_QUEUE_WRITE_CYCLES:sa=2000000,INST_QUEUE_WRITES:sa=2000000,I
NST_RETIRED.ANY:sa=2000000,ITLB_MISS_RETIRED:sa=200000,ITLB_MISSES.ANY:sa=200000,ITLB_MI
SSES.WALK_COMPLETED:sa=200000,L2_RQSTS.IFETCH_HIT:sa=20000,L2_RQSTS.IFETCH_MISS:sa=20000
,MACHINE_CLEARS.CYCLES:sa=20000,MACHINE_CLEARS.MEM_ORDER:sa=20000,MACHINE_CLEARS.SMC:sa=
20000,MEM_LOAD_RETIRED.L2_HIT:sa=200000,MEM_LOAD_RETIRED.LLC_MISS:sa=10000,MEM_LOAD_RETI
RED.LLC_UNSHARED_HIT:sa=40000,MEM_UNCORE_RETIRED.LOCAL_DRAM:sa=10000,MEM_UNCORE_RETIRED.
REMOTE_DRAM:sa=20000,OFFCORE_RESPONSE_0.DATA_IN.LOCAL_CACHE:sa=100000,OFFCORE_RESPONSE_0
.DATA_IN.LOCAL_DRAM:sa=100000,OFFCORE_RESPONSE_0.DATA_IN.REMOTE_DRAM:sa=100000,OFFCORE_R
ESPONSE_0.DEMAND_IFETCH.LOCAL_CACHE:sa=10000,OFFCORE_RESPONSE_0.DEMAND_IFETCH.LOCAL_DRAM
:sa=10000,OFFCORE_RESPONSE_0.DEMAND_IFETCH.REMOTE_DRAM:sa=10000,PARTIAL_ADDRESS_ALIAS:sa
=200000,RESOURCE_STALLS.ANY:sa=2000000,UOPS_DECODED.MS_CYCLES_ACTIVE:sa=2000000,UOPS_DEC
ODED.STALL_CYCLES:sa=2000000,UOPS_ISSUED.ANY:sa=2000000,UOPS_ISSUED.CORE_STALL_CYCLES:sa
=2000000,UOPS_ISSUED.STALL_CYCLES:sa=2000000,UOPS_RETIRED.ANY:sa=2000000,UOPS_RETIRED.RE
TIRE_SLOTS:sa=2000000,UOPS_RETIRED.STALL_CYCLES:sa=2000000 -em dts=100 -out
$OUTPUT_PATH/ATLAS-$EVENTS-GEN/atlas-$EVENTS-ev-gen -app time -args "Evgen_trf.py --
omitvalidation=testEventMinMax ecmEnergy=7000. runNumber=105568 firstEvent=1
maxEvents=$EVENTS randomSeed=1324354657 jobConfig=MC9.105568.ttbar_Pythia.py
outputEvgenFile=pythia-$EVENTS.pool.root" > $OUTPUT_PATH/ATLAS-$EVENTS-GEN/atlas-
$EVENTS-ev-gen.log 2> $OUTPUT_PATH/ATLAS-$EVENTS-GEN/atlas-$EVENTS-ev-gen-err.log
```

SEP's memory usage and cpu time was measured while monitoring ATLAS simulation stage:
VIRT RES SHR
230MB 11MB 1400KB
real 67m3.966s
user 64m38.142s
sys 1m33.279s

3.3 Monitoring the frameworks

Performance monitoring measurements were carried out for the experiments, using the following stages/steps:

- ALICE 64-bit – 2 stages: simulation and reconstruction
- ATLAS 64-bit – 4 stages: generation, simulation, digitization, reconstruction
- ATLAS 32-bit (used in production, because the 64-bit version has higher memory requirements) – 4 stages: generation, simulation, digitization, reconstruction --without analysis of the results
- CMS 64-bit – 2 steps:
  - step 1→ generation, simulation, digitization (run in production)
  - step 2 → reconstruction
- LHCb 64-bit (used in production)– 3 stages:
  - Gauss → generation & simulation
  - Boole → digitization
  - Brunel → reconstruction

The machine on which they were carried has the following specifications:

Processor: Intel(R) Xeon(R) CPU L5520 @ 2.27GHz
L2 Cache Size: 256 KB
L3 Cache size: 8192 KB

Bash scripts were developed for each framework in order to simplify the process of running and monitoring the frameworks. Python scripts were created for editing the configuration/job option files for modifying easily the number of events. This is done in order to obtain different runtimes for the frameworks.

The measurements were carried for 1 event, 20-30 minutes simulation runtime, approximately 1 hour simulation runtime and approximately 4 hours simulation runtime and they include the initialization part of

the frameworks, using the SEP command line presented in section 3.3. The measurements were done without pinning the process to a certain core.

3.4 Results

The software frameworks used for performance monitoring were ALICE, ATLAS, CMS and LHCb. In this section, a brief an very initial discussion of the results is presented.

The measurements were collected as tb5 files (data sampling files produced by SEP). Generally, these files are very large, as can be seen from the total amount of data that was collected:

- Data collected for LHCb 29GB
- Data collected for CMS 17GB
- Data collected for ATLAS 32GB
- Data collected for ALICE 25GB

For Gauss run with 50 events, the tb5 file has 1.2GB. For 50 events simulation with ATLAS, the tb5 file has 3.5G.

The files were manually exported to CSV files with function granularity and source granularity, sorted by thread clock cycles. A detailed description of the files can be found in the Appendix.

3.4.1 ALICE

For the ALICE software framework, the performance monitoring tasks were carried for the two stages (simulation and reconstruction).For the simulation stage, it was observed that the framework spends almost 40% of the time in the following 2 functions: `AliCheb3DCalc::ChebEval1D(float, float const*, int)` and `AliCheb3DCalc::Eval(float*) const.` A significant part - 3% - is devoted to dynamic casting in C++. Also, the mathematical functions have an important percentage - 10 % - (`cos, __ieee754_log, sin , __ieee754_exp, __ieee754_atan2, __ieee754_pow`).
In the reconstruction stage, the management of data (`memset, inflate_fast, _int_malloc, _int_free, malloc_consolidate`) takes approximately 20% of the time, but since this is a data intensive process this is not surprising. The top function is memset, with approximately 13 % of the time.

3.4.2 ATLAS

The performance monitoring tasks were carried for the 4 execution stages in ATLAS 64-bit. In the simulation phase, the numerical functions are taking 10%. In the reconstruction stage, the Linux kernel and `do_lookup_x` are on top, with 7.33% and 4.5 % respectively. Also, the `new` and `delete` operators hold a significant percentage. From the measurements, it results that the profile is a standard one.

3.4.3 CMS

For CMS, the performance monitoring tasks were carried for the two steps. For the first step in CMS, it can be observed that memory management functions are on top. Also, a significant part is accounted for the mathematical functions (__ieee754_log, __ieee754_exp , __ieee754_atan2), along with the Linux kernel. In the second step (reconstruction), the top functions are memory management functions and internal Python functions.

3.4.4 LHCb

The performance monitoring tasks were carried for Gauss, Boole and Brunel. In the generation and simulation phase (Gauss), the following functions are on top: `G4LogicalBorderSurface::GetSurface(G4VPhysicalVolume const*, G4VPhysicalVolume const*)` and `CLHEP::RanluxEngine::flat(void)` and `__ieee754_log`. The memory management functions account for 4% of the time and the Linux kernel for 3%. For Boole, the Linux kernel (approximately 5.5 %) and the data management functions `(_int_malloc, _int_free, malloc, memcpy, do_lookup_x, malloc_consolidate, strcmp )` are on top. The memory management functions have 15% of the time.
In the reconstruction phase, the Linux kernel (5%) and the memory management functions (10%) are on top. Also, the mathematical functions, `__ieee754_log` and `__ieee754_exp,` have a significant percentage. The profile is a standard one.

3.5 Performance monitoring with pfmon

The CERN profile had to be reconstructed for pfmon. The following is a proposal for grouping the events in event sets, as required by pfmon in order to be able to monitor more events than the are actual physical counters on the PMU:

Group 1:
- BR_INST_EXEC:DIRECT_NEAR_CALL
- BR_INST_EXEC:INDIRECT_NEAR_CALL
- BR_INST_EXEC:INDIRECT_NON_CALL
- BR_INST_EXEC:NON_CALLS

Group 2:
- BR_INST_RETIRED:ALL_BRANCHES
- BR_INST_RETIRED:NEAR_CALL
- BR_MISP_EXEC:ANY

Group 3:
- ARITH:CYCLES_DIV_BUSY
- DTLB_MISSES:ANY
- INST_DECODED:DEC0

Group 4:
- INST_QUEUE_WRITE_CYCLES
- INST_QUEUE_WRITES
- INST_RETIRED:ANY_P
- CPU_CLK_UNHALTED:THREAD_P

Group 5:
- ITLB_MISS_RETIRED
- ITLB_MISSES:ANY
- ITLB_MISSES:WALK_COMPLETED

Group 6:
- L2_RQSTS:IFETCH_HIT
- L2_RQSTS:IFETCH_MISS
- MEM_UNCORE_RETIRED:LOCAL_DRAM
- MEM_UNCORE_RETIRED:REMOTE_DRAM

Group 7:
- MACHINE_CLEARS:CYCLES
- MACHINE_CLEARS:MEM_ORDER
- MACHINE_CLEARS:SMC

Group 8:
- MEM_LOAD_RETIRED:L2_HIT
- MEM_LOAD_RETIRED:LLC_MISS
- MEM_LOAD_RETIRED:LLC_UNSHARED_HIT

Group 9:

- PARTIAL_ADDRESS_ALIAS
- RESOURCE_STALLS:ANY
- UOPS_ISSUED:ANY
- UOPS_ISSUED:STALL_CYCLES

Group 10:

- UOPS_RETIRED:ANY
- UOPS_RETIRED:RETIRE_SLOTS
- UOPS_RETIRED:STALL_CYCLES

A Python application was developed for allowing pfmon to attach to the process that we want to monitor after the initialization part has finished based on the output of the framework. The application is using the pexpect module and pfmon's option for attaching to a running process - --attach-task.

```
./app.py -h
usage: app.py -opt1 arg1 -opt2 arg2 -opt3 arg3

options:
  -h, --help            show this help message and exit
  -e EXPRESSION, --expression=EXPRESSION
                        regular expression or string - to be embedded within
                        quotes
  -c COMMAND, --command=COMMAND
                        command to be run - to be embedded within quotes
  -p PARAMETERS, --pfmon-opt=PARAMETERS
                        options for pfmon - to be embedded within quotes
```

For ATLAS, the initialization part also goes on during event 1 and a little part of event 2. The output message from the ATLAS software framework that the application should receive as a parameter is

```
AthenaEventLoopMgr   INFO   ===>>>  start processing event #number1, run #number2
number3 events processed so far  <<<===
```

Note that the run number changes and that the event number could also change.

For LHCb applications, the output message that should be given as a parameter is

```
ApplicationMgr       INFO Application Manager Started successfully
```

In reality, the initialization is still taking place while the application is processing the first events. It is recommended that the first 5-10 events should be skipped, meaning that the output message should be

```
ApplicationNameInit           INFO Evt number1,  Run number2,  Nr. in job = 6
```

where ApplicationName can be Gauss, Boole or Brunel.

Example:

```
BrunelInit           INFO Evt [0-9]*,  Run [0-9]*,  Nr. in job = 6
```

For CMS and ALICE, there is yet no information about which output messages from the frameworks should be used for skipping the initialization part.

3.6 Conclusions

In the first part, we describe the steps required for installing and running the 4 major software frameworks for the experiments at CERN - ALICE, ATLAS, CMS and LHCb. The performance monitoring of the frameworks cannot be done without a solid understanding of this part.

The performance monitoring tasks were carried using the sampling mode of SEP. The results were presented briefly in the last part. Further studies should focus on understanding how to use SEP in counting mode.

For *pfmon*, the CERN profile has to be reconstructed completely. Also, running *pfmon* with the CERN profile in sampling mode (with multiplexing) and counting mode should be better understood.

4 Acknowledgements

I would like to thank the people I collaborated with during my stay at CERN, especially my supervisor, Andrzej Nowak. I want to thank Zachary Marshall, Karol Kruzelecki, Alfio Lazzaro, Julien Leduc and Vincenzo Innocente for their help.

Summary

The first part describes the steps required for installing and running the 4 major software frameworks for the experiments at CERN - ALICE, ATLAS, CMS and LHCb. The performance monitoring tasks were carried using the *Intel Performance Tuning Utility* and *pfmon*. The CERN profile provided by David Levinthal was used in the measurements. Several scripts were developed in order to simplify the monitoring process. A Python application was developed in order to allow *pfmon* to attach to the process that was being monitored after the initialization part has finished based on the output of the framework.

5 References

[1] https://twiki.cern.ch/twiki/bin/view/Atlas/WorkBook

[2] https://twiki.cern.ch/twiki/bin/view/Atlas/RegularComputingTutorial#Running_detector_simulation_and

[3] https://twiki.cern.ch/twiki/bin/view/LHCb/SimDigiReconTutorial#Generate_digitize_and_reconstruc

[4] https://twiki.cern.ch/twiki/bin/view/LHCb/SetupProject

[5] http://cmssdt.cern.ch/SDT/html/showIB.html

[6] http://aliweb.cern.ch/Offline/AliRoot/Manual.html

[7] http://perfmon2.sourceforge.net/pfmon_usersguide.html#sampopt

[8] Intel® Performance Tuning Utility – User guide

[9] https://twiki.cern.ch/twiki/bin/view/Openlab/Perfmon2

6 Appendix

5.1 ATLAS

The general steps are as follows:

1. Create a cmthome directory in your home directory:

```
cd $HOME
mkdir cmthome
cd cmthome
```

2. Create a login requirements file

```
set   CMTSITE  CERN
set   SITEROOT /afs/cern.ch
macro ATLAS_DIST_AREA ${SITEROOT}/atlas/software/dist
macro ATLAS_TEST_AREA ${HOME}/scratch1/
use AtlasLogin AtlasLogin-* $(ATLAS_DIST_AREA)
```

The path after `${HOME}` should be changed so that it points to the user's working directory.

3. Set up CMT as follows:

```
source /afs/cern.ch/sw/contrib/CMT/vNrP/mgr/setup.sh
cmt config
```

The first line sets the CMT version.

4. Set up the ATLAS software

```
mkdir scratch1
mkdir /afs/cern.ch/user/d/dpopescu/scratch1/AtlasOffline-15.8.0
cd /afs/cern.ch/user/d/dpopescu/scratch1/AtlasOffline-15.8.0


cd $HOME
source cmthome/setup.sh -tag=15.8.0,64,setup,opt,gcc43
mkdir $TestArea
cd $TestArea
```

It can be seen here that the version used is 15.8.0, composed of 3 numbers, which means that Atlas will be running from the AtlasOffline area. AtlasOffline means the entire software is taken rather than a certain project (group of packages). For modifying the compiling options, we can change from 64-bit to 32-bit, specifying this in the options.

For setting up ATLAS on 32-bit, the following commands can be used:

```
cd /tmp/`whoami`
source /afs/cern.ch/atlas/software/releases/15.8.0/cmtsite/setup.sh -
tag=AtlasProduction,15.8.0.2,32,noTest
source
/afs/cern.ch/atlas/software/releases/15.8.0/AtlasProduction/15.8.0.2/AtlasProductionRunT
ime/cmt/setup.sh
```

The second command has to be run every time the user logs into a new session.

The general steps for running the framework are:

1. Download the job option file

```
get_files -jo MC9.105145.PythiaZmumu.py
```

2. At the end of the file, the following line can be added for modifying the number of events that will be generated:

`evgenConfig.minevents = 100` (number of events)

Default, the number of generated events is 5000.

3. Event generation

```
Evgen_trf.py --omitvalidation=testEventMinMax ecmEnergy=7000. runNumber=105144
firstEvent=1 maxEvents=10 randomSeed=1324354657
jobConfig=MC09JobOptions/MC9.105144.PythiaZee.py outputEvgenFile=pythia.pool.root

Evgen_trf.py –h
```

Arguments:

```
    1 ecmEnergy (float) # Center of mass energy parameter in GeV
    2 runNumber (int) # each run number corresponds to one physics process
    3 firstEvent (int) # number of the first event in the output data file
  [ 4 maxEvents] (int) default=-1 # Maximum number of events to process
    5 randomSeed (int) # random seed for physics generators
    6 jobConfig (list) # jobOptions fragment containing the physics and the
configuration settings
    7 outputEvgenFile (str) # Output file that contains generated events
  [ 8 histogramFile] (str) default='NONE' # Output file that contains histograms.
  [ 9 ntupleFile] (str) default='NONE' # Output file that contains ntuples.
  [10 inputGeneratorFile] (str) default='NONE' # Input file used by the particle
generator to generate events
  [11 EvgenJobOpts] (str) default='NONE' # Tarball containing the DBRelease to use
```

4. Simulation

```
      csc_atlasG4_trf.py inputEvgenFile=pythia.pool.root outputHitsFile=g4hits.pool.root
maxEvents=10 skipEvents=0 randomSeed=1234 geometryVersion=ATLAS-GEO-06-00-00
conditionsTag=OFLCOND-SIM-BS7T-02 physicsList=QGSP_BERT jobConfig=NONE

csc_atlasG4_trf.py  –h
```

Arguments:

```
    1 inputEvgenFile (list) # Input file that contains generated events
    2 outputHitsFile (str) # Output file that contains hits
  [ 3 maxEvents] (int) default=-1 # Maximum number of events to process
    4 skipEvents (int) # Number of events to skip
    5 randomSeed (int) # random seed for simulation
    6 geometryVersion (str) # Geometry Version
  [ 7 physicsList] (str) default='QGSP_BERT' # physics list to be used
  [ 8 jobConfig] (list) default='NONE' # Joboptions file with user settings, in
particular the configuration settings. Default packages:
.,SimuJobTransforms,PyJobTransforms
  [ 9 DBRelease] (str) default='NONE' # Tarball containing the DBRelease to use
  [10 conditionsTag] (str) default='NONE' # IOVDb global tag
  [11 IgnoreConfigError] (bool) default=False # Allow Resilience to catch exceptions
from configuration errors
  [12 AMITag] (str) default='NONE' # Store the AMITag for this step
```

5. Digitization

```
      csc_digi_trf.py inputHitsFile=g4hits.pool.root outputRDOFile=g4digi.pool.root
maxEvents=-1 skipEvents=0 geometryVersion=ATLAS-GEO-06-00-00  conditionsTag=OFLCOND-DR-
BS7T-ANom-06 digiseedoffset1=11 digiseedoffset2=22

csc_digi_trf..py  –h
```

Arguments:

```
    1 inputHitsFile (list) # Input file that contains hits
    2 outputRDOFile (str) # Output file that contains RDO's
  [ 3 maxEvents] (int) default=-1 # Maximum number of events to process
    4 skipEvents (int) # Number of events to skip
    5 geometryVersion (str) # Geometry Version
    6 digiSeedOffset1 (int) # random seed offset for digitization
    7 digiSeedOffset2 (int) # random seed offset for digitization
  [ 8 minbiasHitsFile] (list) default='NONE' # input hits for pile-up
  [ 9 cavernHitsFile] (list) default='NONE' # input hits for cavern background
  [10 jobConfig] (list) default='NONE' # Joboptions file with user settings, in
particular the configuration settings. Default packages:
.,SimuJobTransforms,PyJobTransforms
  [11 DBRelease] (str) default='NONE' # Tarball containing the DBRelease to use
```

```
  [12 digiRndmSvc] (str) default='AtRanluxGenSvc' # random number service to use for
digitization - AtRndmGenSvc uses Ranecu, AtRanluxGenSvc used Ranlux64
  [13 samplingFractionDbTag] (str) default='QGSP_BERT' # liquid argon calorimeter
sampling fraction data base tag, passed on in jobOptions to LArfSamplG4Phys
  [14 triggerConfig] (str) default='default' # Configuration string to use for TrigT1
and HLT. Set to 'NONE' to switch off trigger,
    and set to 'DEFAULT' to use the default of the used release.
  [15 beamHaloHitsFile] (list) default='NONE' # input hits for beam halo: side A - 98
percent, side C - 2 percent
  [16 beamGasHitsFile] (list) default='NONE' # input hits for beam gas, already mixed
and flipped C, H and O
  [17 doAllNoise] (str) default='NONE' # override noise simulation flags in all
subdetectors
  [18 AddCaloDigi] (bool) default=False # True/False: Save CaloDigits as well, not just
RawChannels
  [19 conditionsTag] (str) default='NONE' # IOVDb global tag
  [20 IgnoreConfigError] (bool) default=False # Allow Resilience to catch exceptions
from configuration errors
  [21 AMITag] (str) default='NONE' # Store the AMITag for this step
  [22 DataRunNumber] (int) default='-1' # Override existing run number with this value
  [23 SDMinbiasHitsFile] (list) default='NONE' # input hits for single-diffractive
minbias
  [24 DDMinbiasHitsFile] (list) default='NONE' # input hits for double-diffractive
minbias
```

6.Reconstruction

```
Reco_trf.py  inputRDOFile=g4digi.pool.root outputESDFile=esd.pool.root maxEvents=-
1 skipEvents=0 geometryVersion=ATLAS-GEO-06-00-00 conditionsTag=OFLCOND-DR-BS7T-ANom-06
```

5.2 ALICE

1. Set the installation path modifying the variable ALICE in `alice_vars.sh` (default: $HOME/ALICE).

2. Run `alice_install.sh`. The script will install the framework. Syntax:

```
alice_install.sh [-j number_of_jobs]
```

-j number_of_jobs: Specifies the number of jobs to run simultaneously during the compilation (default: 1)

For running Alice, use the script `alice_run.sh`. Syntax:

```
alice_run.sh -n number_of_events [[-p] || [-c pfmon_command]] [-o output_path]
```

-n number_of_events: The number of events that will be generated.
-p: Activate profiling with pfmon (default flags).
-c pfmon_command: Activate profiling with pfmon specifying the pfmon command (use " ").
-o output_path: The path where you want to generate the output files.

5.3 CMS

1.Download the necessary archives:

```
cd /disk1/cmssw
wget http://cmsrep.cern.ch/cmssw/dl/20100525/SITECONF.tgz
wget http://cmsrep.cern.ch/cmssw/dl/20100525/common.tgz
wget http://cmsrep.cern.ch/cmssw/dl/20100525/frontier.tgz
wget http://cmsrep.cern.ch/cmssw/dl/20100525/slc5_amd64_gcc434_cms.tbz
wget http://cmsrep.cern.ch/cmssw/dl/20100525/slc5_amd64_gcc434_external.tbz
wget http://cmsrep.cern.ch/cmssw/dl/20100525/slc5_amd64_gcc434_lcg.tgz
wget http://cmsrep.cern.ch/cmssw/dl/20100525/slc5_amd64_gcc434_other.tgz
```

```
gtar xzf 20100525/SITECONF.tgz
gtar xzf 20100525/common.tgz
gtar xzf 20100525/frontier.tgz
gtar xjf slc5_amd64_gcc434_cms.tbz
gtar xjf slc5_amd64_gcc434_external.tbz
gtar xzf slc5_amd64_gcc434_lcg.tgz
gtar xzf slc5_amd64_gcc434_other.tgz

perl -p -i -e 's/dbsfrontier.cern.ch/128.142.202.212/' frontier/frontier-
cache/squid/etc/squid.conf.frontierdefault
perl -p -i -e 's/frontier.cern.ch/128.142.202.212/' frontier/frontier-
cache/squid/etc/squid.conf.frontierdefault
```

2. The following file `frontier/frontier-cache/squid/etc/squid.conf.frontierdefault` should be updated:

```
cache_mgr user
cache_effective_user user
cache_effective_group user
```

The first two should be changed to the unix userid you are using for the software/frontier installation and the 3rd to the corresponding groupid.

3. Edit `frontier/frontier-cache/squid/etc/customize.sh` by adding the following three lines:

```
setoption("offline_mode", "on")
commentout("^snmp_access allow")
commentout("^acl HOST_MONITOR")
```

4. Start the frontier web squid. It will be visible on localhost port 3128 and given the SITECONF configuration normal CMS jobs will contact it as a proxy in order to obtain conditions data. First run a script that updates some additional configuration files based on the changes you made above:

```
/disk1/cmssw/frontier/frontier-cache/utils/bin/fn-local-squid.sh
```

Start the squid:

```
/disk1/cmssw/frontier/frontier-cache/utils/bin/fn-local-squid.sh start
```

5. Set up the general shell environment needed to access the CMS software.

```
# tcsh variant of CMS software environment
setenv SCRAM_ARCH slc5_amd64_gcc434
setenv VO_CMS_SW_DIR /disk1/cmssw
source $VO_CMS_SW_DIR/cmsset_default.csh

# bash variant of CMS software environment
export SCRAM_ARCH=slc5_amd64_gcc434
export VO_CMS_SW_DIR=/disk1/cmssw
source $VO_CMS_SW_DIR/cmsset_default.sh
```

This has to be done every time the user logs into a new session.

6. Create a SCRAM development area based on a particular release from the CMS software installation and set up additional specifics in the shell environment to use that particular release.

```
mkdir -p /disk1/cmssw/work
cd /disk1/cmssw/work
cmsrel CMSSW_3_7_0_pre5
cd CMSSW_3_7_0_pre5/src
cmsenv
```

The 'cmsrel' command is the one that actually creates the SCRAM development area and it only needs to be issued once. If you create a SCRAM development area and later (in a new or an additional shell) want to set things up to use this pre-existing area, you need to set up the general CMS software environment above and then do:

```
cd /disk1/cmssw/work
cd CMSSW_3_7_0_pre5/src
cmsenv


cmsDriver.py -h

Usage: cmsDriver.py <TYPE> [options].
Example:

cmsDriver.py reco -s RAW2DIGI,RECO --conditions
FrontierConditions_GlobalTag,STARTUP_V4::All --eventcontent RECOSIM


Options:
  -h, --help            show this help message and exit
  -s STEP, --step=STEP  The desired step. The possible values are: GEN,SIM,DIG
                        I,L1,DIGI2RAW,HLT,RAW2DIGI,RECO,POSTRECO,DQM,ALCA,VALI
                        DATION,HARVESTING, NONE or ALL.
  --conditions=CONDITIONS
                        What conditions to use. Default are frontier
                        conditions 'STARTUP_V4::All'
                        (FrontierConditions_GlobalTag,STARTUP_V4::All)
  --eventcontent=EVENTCONTENT
                        What event content to write out. Default=FEVTDEBUG, or
                        FEVT (for cosmics)
  --filein=FILEIN       The infile name.
  --fileout=FILEOUT     The outfile name. If absent a default value is
                        assigned
  --filetype=FILETYPE   The type of the infile (EDM, LHE or MCDB).
  -n NUMBER, --number=NUMBER
                        The number of events. The default is 1.
  --mc                  Specify that simulation is to be processed (default =
                        guess based on options
  --data                Specify that data is to be processed (default = guess
                        based on options
  --no_exec             Do not exec cmsRun. Just prepare the python config
                        file.

  ===============
  Expert Options:
    Caution: please use only if you know what you are doing.

    --beamspot=BEAMSPOT
                        What beam spot to use (from
                        Configuration/StandardSequences). Default depends on
                        scenario
    --customise=CUSTOMISATION_FILE
                        Specify the file where the code to modify the process
                        object is stored.
    --datatier=DATATIER
                        What data tier to use.
    --dirin=DIRIN       The infile directory.
    --dirout=DIROUT     The outfile directory.
    --filtername=FILTERNAME
                        What filter name to specify in output module
    --geometry=GEOMETRY
                        What geometry to use (from
                        Configuration/StandardSequences). Default=Ideal
    --magField=MAGFIELD
                        What magnetic field to use (from
                        Configuration/StandardSequences).
    --no_output         Do not write anything to disk. This is for
                        benchmarking purposes.
    --oneoutput         use only one output module
    --prefix=PREFIX     Specify a prefix to the cmsRun command.
    --relval=RELVAL     Set total number of events and events per job.
    --dump_python       Dump the config file in python and do a full expansion
                        of imports.
```

```
--dump_DSetName     Dump the primary datasetname.
--pileup=PILEUP     What pileup config to use. Default=NoPileUp.
--datamix=DATAMIX   What datamix config to use. Default=DataOnSim.
--gflash            Run the FULL SIM using the GFlash parameterization.
--himix             Run the Heavy Ions signal mixing.
--python_filename=PYTHON_FILENAME
                    Change the name of the created config file
--secondfilein=SECONDFILEIN
                    The secondary infile name.for the two-file solution.
                    Default is no file
--writeraw          In addition to the nominal output, write a file with
                    just raw
--processName=NAME  set process name explicitly
--triggerResultsProcess=TRIGGERRESULTSPROCESS
                    for splitting jobs specify from which process to take
                    edm::TriggerResults
--scenario=SCENARIO
                    Select scenario overriding standard settings
                    (available:['pp', 'cosmics', 'nocoll', 'HeavyIons'])
--harvesting=HARVESTING
                    What harvesting to use (from
                    Configuration/StandardSequences). Default=AtRunEnd
--particle_table=PARTICLETABLE
                    Which particle properties table is loaded.
                    Default=pythia
--dbsquery=DBSQUERY
                    Allow to define the source.fileNames from the dbs
                    search command
```

5.4 LHCb

The LHCb software environment is set up by sourcing the script LbLogin.[c]sh

```
source /afs/cern.ch/lhcb/software/releases/LBSCRIPTS/prod/InstallArea/scripts/LbLogin.sh
--no-cache
```

or

```
source
/afs/cern.ch/lhcb/software/releases/LBSCRIPTS/prod/InstallArea/scripts/LbLogin.csh --no-
cache
```

The command has to be run every time the user logs into a new session.

`SetupProject ProjectName` – it will take the latest available version.

If the user wants to modify options or write new packages, he will need to create a user local project as shown below and check out the necessary packages.

Gauss - Simulation

1.The following commands make an executable version of Gauss available in the user's lxplus area

```
SetupProject Gauss vNrP --build-env
getpack Sim/Gauss vNrP
cd Sim/Gauss/cmt
cmt make
SetupProject Gauss vNrP
cd ../options
```

Note : SetupProject Package vNrP --build-env needs to be run only once to make the directory. However, SetupProject Package vNrP must be run every time package is being used.

2. Edit Gauss-Job.py to simulate the required events

For changing the number of events the following line should be edited:

```
nEvts = 5
```

Note: For changing the name of the output file, the following line should be edited:

```
idFile = 'FileName'
```

If this line is not changed, Gauss will make a name based on event type, number of events and the date.

3. The command gaudirun.py must be used for running Gauss, specifying the appropriate arguments to the command.

Example: `gaudirun.py $GAUSSOPTS/Gauss-MC09.py $DECFILESROOT/options/11144101.opts $GAUSSOPTS/Gauss-Job.py`

```
      gaudirun.py $GAUSSOPTS/Gauss-DEV.py $DECFILESROOT/options/30000000.opts
$LBPYTHIAROOT/options/Pythia.opts $GAUSSOPTS/Gauss-Job.py
```

Boole – Digitization

1. The following commands make an executable version of Boole available in the user's lxplus area

```
SetupProject Boole vNrP --build-env
getpack Digi/Boole vNrP
cd Digi/Boole/cmt
cmt make
SetupProject Boole vNrP
cd ../options
```

2. Edit MC09-Files.py to setup Boole to digitize the events generated earlier.

Change the input data file to the FileName.sim file generated earlier in Gauss.

```
datasetname = 'FileName'
EventSelector().Input = ["DATAFILE='PFN:$HOME/cmtuser/Gauss_vNrP/Sim/Gauss/options/" +
datasetName + ".sim' TYP='POOL_ROOTTREE' OPT='READ'
```

3. Run Boole with `gaudirun.py $BOOLEOPTS/Boole-MC09-WithTruth.py $BOOLEOPTS/MC09-Files.py`

Brunel – Reconstruction

1. The following commands make an executable version of Brunel available in the user's lxplus area

```
SetupProject Brunel vNrP --build-env
getpack Reco/Brunel vNrP
cd Reco/Brunel/cmt
cmt make
SetupProject Brunel vNrP
cd ../options
```

2. In the options directory set the input file to the output produced by Boole, i.e.FileName.digi, in MC09-Files.py. Note the file name to change is the last one which is by default "30000000-100ev-20090407-MC09" and the EventSelector line just underneath this.

3. Run Brunel with `gaudirun.py $BRUNELOPTS/Brunel-MC09-WithTruth.py $BRUNELOPTS/MC09-Files.py`

5.5 CERN profile

ARITH.CYCLES_DIV_BUSY:sa=2000000,
BR_INST_EXEC.DIRECT_NEAR_CALL:sa=20000,
BR_INST_EXEC.INDIRECT_NEAR_CALL:sa=20000,
BR_INST_EXEC.INDIRECT_NON_CALL:sa=20000,
BR_INST_EXEC.NON_CALLS:sa=200000,
BR_INST_RETIRED.ALL_BRANCHES:sa=200000,
BR_INST_RETIRED.NEAR_CALL:sa=4000,
BR_MISP_EXEC.ANY:sa=20000,
CPU_CLK_UNHALTED.THREAD:sa=2000000,
DTLB_MISSES.ANY:sa=200000,
INST_DECODED.DEC0:sa=2000000,
INST_QUEUE_WRITE_CYCLES:sa=2000000,
INST_QUEUE_WRITES:sa=2000000,
INST_RETIRED.ANY:sa=2000000,
ITLB_MISS_RETIRED:sa=200000,
ITLB_MISSES.ANY:sa=200000,
ITLB_MISSES.WALK_COMPLETED:sa=200000,
L2_RQSTS.IFETCH_HIT:sa=20000,
L2_RQSTS.IFETCH_MISS:sa=20000,
MACHINE_CLEARS.CYCLES:sa=20000,
MACHINE_CLEARS.MEM_ORDER:sa=20000,
MACHINE_CLEARS.SMC:sa=20000,
MEM_LOAD_RETIRED.L2_HIT:sa=200000,
MEM_LOAD_RETIRED.LLC_MISS:sa=10000,
MEM_LOAD_RETIRED.LLC_UNSHARED_HIT:sa=40000,
MEM_UNCORE_RETIRED.LOCAL_DRAM:sa=10000,
MEM_UNCORE_RETIRED.REMOTE_DRAM:sa=20000,
OFFCORE_RESPONSE_0.DATA_IN.LOCAL_CACHE:sa=100000,
OFFCORE_RESPONSE_0.DATA_IN.LOCAL_DRAM:sa=100000,
OFFCORE_RESPONSE_0.DATA_IN.REMOTE_DRAM:sa=100000,
OFFCORE_RESPONSE_0.DEMAND_IFETCH.LOCAL_CACHE:sa=10000,
OFFCORE_RESPONSE_0.DEMAND_IFETCH.LOCAL_DRAM:sa=10000,
OFFCORE_RESPONSE_0.DEMAND_IFETCH.REMOTE_DRAM:sa=10000,
PARTIAL_ADDRESS_ALIAS:sa=200000,
RESOURCE_STALLS.ANY:sa=2000000,
UOPS_DECODED.MS_CYCLES_ACTIVE:sa=2000000,
UOPS_DECODED.STALL_CYCLES:sa=2000000,
UOPS_ISSUED.ANY:sa=2000000,
UOPS_ISSUED.CORE_STALL_CYCLES:sa=2000000,
UOPS_ISSUED.STALL_CYCLES:sa=2000000,
UOPS_RETIRED.ANY:sa=2000000,
UOPS_RETIRED.RETIRE_SLOTS:sa=2000000,
UOPS_RETIRED.STALL_CYCLES:sa=2000000

5.6 Listing of tb5 and CSV files

5.6.1 ALICE

The following measurements were carried for the ALICE experiment:

1. 1 event :
    - simulation –
        - Tb5 file name: /data1/dpopescu/alice_framework/Alice1ev-sim/alice1ev-sim.tb5
        - CSV file name: /data1/dpopescu/CSV/ALICE/ ALICE-1ev-sim-f.csv – function granularity
          /data1/dpopescu/CSV/ALICE/ ALICE-1ev-sim-s.csv – source granularity
        - Duration: 344.19s
    - reconstruction –
        - Tb5 file name: /data1/dpopescu/alice_framework/Alice1ev-reco/alice1ev-reco.tb5
        - CSV file name: /data1/dpopescu/CSV/ALICE/ ALICE-1ev-reco-f.csv – function granularity
          /data1/dpopescu/CSV/ALICE/ ALICE-1ev-reco-s.csv – source granularity
        - Duration: 26.04s
2. 10 events
    - simulation –
        - Tb5 file name: /data1/dpopescu/alice_framework/Alice10evsim/alice10evsim.tb5
        - CSV file name: /data1/dpopescu/CSV/ALICE/ ALICE-10ev-sim-f.csv – function granularity
          /data1/dpopescu/CSV/ALICE/ ALICE-10ev-sim-s.csv – source granularity
        - Duration: 1436.39s
    - reconstruction –
        - Tb5 file name: /data1/dpopescu/alice_framework/Alice10evrec/alice10evrec.tb5
        - CSV file name: /data1/dpopescu/CSV/ALICE/ ALICE-10ev-reco-f.csv – function granularity
          /data1/dpopescu/CSV/ALICE/ ALICE-10ev-reco-s.csv – source granularity
        - Duration: 44.78s

3. 30 events
    - simulation –
        - Tb5 file name: /data1/dpopescu/alice_framework/Alice30evsim/alice30evsim.tb5
        - CSV file name: /data1/dpopescu/CSV/ALICE/ ALICE-30ev-sim-f.csv – function granularity
          /data1/dpopescu/CSV/ALICE/ ALICE-30ev-sim-s.csv – source granularity
        - Duration: 4800.62s
    - reconstruction –
        - Tb5 file name: /data1/dpopescu/alice_framework/Alice30evreco/alice30evreco.tb5
        - CSV file name: /data1/dpopescu/CSV/ALICE/ ALICE-30ev-reco-f.csv – function granularity
          /data1/dpopescu/CSV/ALICE/ ALICE-30ev-reco-s.csv – source granularity
        - Duration: 142.16s

4. 100 events
    - simulation –
        - Tb5 file name: /data1/dpopescu/alice_framework/Alice10evsim/alice100ev-sim.tb5
        - CSV file name: /data1/dpopescu/CSV/ALICE/ ALICE-100ev-sim-f.csv – function granularity
          /data1/dpopescu/CSV/ALICE/ ALICE-100ev-sim-s.csv – source granularity

- Duration: 6101.88s

- reconstruction –
  - Tb5 file name: /data1/dpopescu/alice_framework/Alice100ev-reco/alice100ev-reco.tb5
  - CSV file name: /data1/dpopescu/CSV/ALICE/ ALICE-100ev-reco-f.csv – function granularity
    /data1/dpopescu/CSV/ALICE/ ALICE-100ev-reco-s.csv – source granularity
  - Duration: 402.58s

The script alice_run_sep.sh was used for running and monitoring ALICE.

The script alice_run_sep_taskset.sh provides log files, automatic directory organization and pinning (taskset). `Usage: alice_run_sep_taskset.sh -n number_of_events -m factor -o output_path`. The –m option refers to the possibility of multiplying the sampling periods for the events with a factor of 10 or 100. The script alice_run_collection.sh can be used for running ALICE with 1 event, 10 events, 30 events and 100 events, having as output path the current directory.

The scripts can be found in /data1/dpopescu/alice_framework.

5.6.2 ATLAS

64 bit – ttbar example

1. 1 event :
   - Generation –
     - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS1ev-gen/atlas1ev-gen.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas1ev-gen-f.csv – function granularity
       /data1/oplaatl2/ATLAS/CSV/atlas1ev-gen-s.csv – source granularity
     - Duration: 34.57s
   - Simulation –
     - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS1ev-sim/atlas1ev-sim.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas1ev-sim-f.csv – function granularity
       /data1/oplaatl2/ATLAS/CSV/atlas1ev-sim-s.csv – source granularity
     - Duration: 653.41s
   - Digitization -
     - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS1ev-gen/atlas1ev-digi.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas1ev-digi-f.csv – function granularity
       /data1/oplaatl2/ATLAS/CSV/atlas1ev-digi-s.csv – source granularity
     - Duration: 234.38s
   - Reconstruction –
     - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS1ev-reco/atlas1ev-reco.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas1ev-reco-f.csv – function granularity
       /data1/oplaatl2/ATLAS/CSV/atlas1ev-reco-s.csv – source granularity
     - Duration: 579.26s
2. 5 events
   - Generation –
     - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS5ev-gen/atlas5ev-gen.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas5ev-gen-f.csv – function granularity
       /data1/oplaatl2/ATLAS/CSV/atlas5ev-gen-s.csv – source granularity
     - Duration: 51.03s

- Simulation –
  - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS5ev-sim/atlas5ev-sim.tb5
  - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas5ev-sim-f.csv – function granularity
    /data1/oplaatl2/ATLAS/CSV/atlas5ev-sim-s.csv – source granularity
  - Duration: 2030.89s
- Digitization -
  - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS5ev-gen/atlas5ev-digi.tb5
  - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas5ev-digi-f.csv – function granularity
    /data1/oplaatl2/ATLAS/CSV/atlas5ev-digi-s.csv – source granularity
  - Duration: 270.49s
- Reconstruction –
  - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS5ev-reco/atlas5ev-reco.tb5
  - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas5ev-reco-f.csv – function granularity
    /data1/oplaatl2/ATLAS/CSV/atlas5ev-reco-s.csv – source granularity
  - Duration: 619.51s

3. 10 events
   - Generation –
     - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS10ev-gen/atlas10ev-gen.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas10ev-gen-f.csv – function granularity
       /data1/oplaatl2/ATLAS/CSV/atlas10ev-gen-s.csv – source granularity
     - Duration: 41.03s
   - Simulation –
     - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS10ev-sim/atlas10ev-sim.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas10ev-sim-f.csv – function granularity
       /data1/oplaatl2/ATLAS/CSV/atlas10ev-sim-s.csv – source granularity
     - Duration: 3875.63s
   - Digitization -
     - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS10ev-gen/atlas10ev-digi.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas10ev-digi-f.csv – function granularity
       /data1/oplaatl2/ATLAS/CSV/atlas10ev-digi-s.csv – source granularity
     - Duration: 295.36s
   - Reconstruction –
     - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS10ev-reco/atlas10ev-reco.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas5ev-reco-f.csv – function granularity
       /data1/oplaatl2/ATLAS/CSV/atlas5ev-reco-s.csv – source granularity
     - Duration: 692.14s

4. 50 events
   - Generation –
     - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS50ev-gen/atlas50ev-gen.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas50ev-gen-f.csv – function granularity

/data1/oplaatl2/ATLAS/CSV/atlas50ev-gen-s.csv – source granularity
    - Duration: 41.03s
- Simulation –
    - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS50ev-sim/atlas50ev-sim.tb5
    - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas50ev-sim-f.csv – function granularity
      /data1/oplaatl2/ATLAS/CSV/atlas50ev-sim-s.csv – source granularity
    - Duration: 3875.63s
- Digitization -
    - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS50ev-gen/atlas50ev-digi.tb5
    - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas50ev-digi-f.csv – function granularity
      /data1/oplaatl2/ATLAS/CSV/atlas50ev-digi-s.csv – source granularity
    - Duration: 295.36s
- Reconstruction –
    - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS50ev-reco/atlas50ev-reco.tb5
    - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas50ev-reco-f.csv – function granularity
      /data1/oplaatl2/ATLAS/CSV/atlas50ev-reco-s.csv – source granularity
    - Duration: 692.14s

The following job option file was used: MC9.105568.ttbar_Pythia.py. The Python script editAtlas.py is used for editing the job option file. This script is called from the bash script runAtlas1.sh, which can be used with the following options: `Usage: runAtlas1.sh -n number_of_events -m factor -o output_path.` It provides log files, automatic directory organization and pinning (taskset). The –m option refers to the possibility of multiplying the sampling periods for the events with a factor of 10 or 100. . The script run_atlas_ex1.sh can be used for running ATLAS with 1 event, 5 events, 10 events and 50 events, having as output path the current directory. The scripts can be found in the directory `/afs/cern.ch/user/o/oplaatl2/scratch1/AtlasOffline-15.8.0/run`.Before running the scripts, the environment has to be set up as described in the Appendix's section dedicated to ATLAS.

64 bit – zmumu example

1. 1 event :
    - Generation –
        - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS1ev-gen-zmumu/atlas1ev-gen-zmumu.tb5
        - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas1ev-gen-zmumu-f.csv – function granularity
          /data1/oplaatl2/ATLAS/CSV/atlas1ev-gen-zmumu-s.csv – source granularity
        - Duration: 41.25s
    - Simulation –
        - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS1ev-sim-zmumu/atlas1ev-sim-zmumu.tb5
        - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas1ev-sim-zmumu-f.csv – function granularity
          /data1/oplaatl2/ATLAS/CSV/atlas1ev-sim-zmumu-s.csv – source granularity

- Duration: 533.29s
- Digitization -
  - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS1ev-gen-zmumu/atlas1ev-digi-zmumu.tb5
  - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas1ev-digi-zmumu-f.csv – function granularity
    /data1/oplaatl2/ATLAS/CSV/atlas1ev-digi-zmumu-s.csv – source granularity
  - Duration: 237.93s
- Reconstruction –
  - Tb5 file name: /data1/oplaatl2/ATLAS/ATLAS1ev-reco-zmumu/atlas1ev-reco-zmumu.tb5
  - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas1ev-reco-zmumu-f.csv – function granularity
    /data1/oplaatl2/ATLAS/CSV/atlas1ev-reco-zmumu-s.csv – source granularity
  - Duration: 646.17s

2. 10 events
   - Generation –
     - Tb5 file name: /data1/oplaatl2/ATLAS/Atlas10ev-gen-zmumu/atlas10ev-gen-zmumu.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas10ev-gen-zmumu-f.csv – function granularity
       /data1/oplaatl2/ATLAS/CSV/atlas10ev-gen-zmumu-s.csv – source granularity
     - Duration: 44.77s
   - Simulation –
     - Tb5 file name: /data1/oplaatl2/ATLAS/Atlas10ev-sim-zmumu/atlas10ev-sim-zmumu.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas10ev-sim-zmumu-f.csv – function granularity
       /data1/oplaatl2/ATLAS/CSV/atlas10ev-sim-zmumu-s.csv – source granularity
     - Duration: 2230.6s
   - Digitization -
     - Tb5 file name: /data1/oplaatl2/ATLAS/Atlas10ev-gen-zmumu/atlas10ev-digi-zmumu.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas10ev-digi-zmumu-f.csv – function granularity
       /data1/oplaatl2/ATLAS/CSV/atlas10ev-digi-zmumu-s.csv – source granularity
     - Duration: 291.7s
   - Reconstruction –
     - Tb5 file name: /data1/oplaatl2/ATLAS/Atlas10ev-reco-zmumu/atlas10ev-reco-zmumu.tb5
     - CSV file name: /data1/oplaatl2/ATLAS/CSV/atlas10ev-reco-zmumu-f.csv – function granularity
       /data1/oplaatl2/ATLAS/CSV/atlas10ev-reco-zmumu-s.csv – source granularity
     - Duration: 601.09s

The following job option file was used: MC9.105145.PythiaZmumu.py. The Python script editAtlas.py is used for editing the job option file. This script is called from the bash script

runAtlas1.sh, which can be used with the following options: `Usage: runAtlas2.sh -n number_of_events -m factor -o output_path.` It provides log files, automatic directory organization and pinning (taskset). The –m option refers to the possibility of multiplying the sampling periods for the events with a factor of 10 or 100. The script run_atlas_ex2.sh can be used for running ATLAS with 1 event, 5 events, 10 events and 50 events, having as output path the current directory. The scripts can be found in the directory `/afs/cern.ch/user/o/oplaatl2/scratch1/AtlasOffline-15.8.0/run`.Before running the scripts, the environment has to be set up as described in the Appendix's section dedicated to ATLAS.

5.6.3 CMS

Minbias example

1. 1 event
   - Step 1 (generation, simulation, digitization) –
     - Tb5 file name: /data1/dpopescu/CMS/CMS1ev-s1/cms1ev-s1.tb5
     - CSV file name:/data1/dpopescu/CSV/CMS/ CMS-1ev-minbias-s1-f.csv – function granularity
       /data1/dpopescu/CSV/CMS/ CMS-1ev-minbias-s1-s.csv – source granularity
     - Duration: 214.92s
   - Step 2 (reconstruction) –
     - Tb5 file name: /data1/dpopescu/CMS/CMS1ev-s2/cms1ev-s2.tb5
     - CSV file name: /data1/dpopescu/CSV/CMS/ CMS-1ev-minbias-s2-f.csv – function granularity
       /data1/dpopescu/CSV/CMS/ CMS-1ev-minbias-s2-s.csv – source granularity
     - Duration: 35.85s

2. 100 events
   - Step 1 (generation, simulation, digitization) –
     - Tb5 file name: /data1/dpopescu/CMS/CMS100ev-s1/cms100ev-s1.tb5
     - CSV file name: :/data1/dpopescu/CSV/CMS/ CMS-100ev-minbias-s1-f.csv – function granularity
       /data1/dpopescu/CSV/CMS/ CMS-100ev-minbias-s1-s.csv – source granularity
     - Duration: 1078.6s
   - Step 2 (reconstruction) –
     - Tb5 file name: /data1/dpopescu/CMS/CMS100ev-s2/cms100ev-s2.tb5
     - CSV file name: :/data1/dpopescu/CSV/CMS/ CMS-100ev-s2-minbias-f.csv – function granularity
       /data1/dpopescu/CSV/CMS/ CMS-100ev-s2-minbias-s.csv – source granularity
     - Duration: 64.98s

3. 300 events
   - Step 1 (generation, simulation, digitization) –
     - Tb5 file name: /data1/dpopescu/CMS/CMS300ev-s1/cms300ev-s1.tb5
     - CSV file name:/data1/dpopescu/CSV/CMS/ CMS-300ev-minbias-s1-f.csv – function granularity
       /data1/dpopescu/CSV/CMS/ CMS-300ev-minbias-s1-s.csv – source granularity
     - Duration: 3533.78s

- Step 2 (reconstruction) –
  - Tb5 file name: /data1/dpopescu/CMS/CMS300ev-s2/cms300ev-s2.tb5
  - CSV file name: /data1/dpopescu/CSV/CMS/ CMS-300ev-minbias-s2-f.csv – function granularity
    /data1/dpopescu/CSV/CMS/ CMS-300ev-minbias-s2-s.csv – source granularity
  - Duration: 136.56s

Qcd example- 100 events

- Step 1 (generation, simulation, digitization) –
  - Tb5 file name: /data1/dpopescu/CMS/CMS100ev-s1-qcd/qcdpt100-s1.tb5
  - CSV file name: /data1/dpopescu/CSV/CMS/CMS-100ev-qcd-s1-f.csv – function granularity
    /data1/dpopescu/CSV/CMS/CMS-100ev-qcd-s1-s.csv – function granularity
  - Duration: 4923.85s
- Step 2 (reconstruction) –
  - Tb5 file name: /data1/dpopescu/CMS/CMS-100ev-qcd-s2/cms-100ev-qcd-s2.tb5
  - CSV file name: /data1/dpopescu/CSV/CMS/CMS-100ev-qcd-s2-f.csv – function granularity
    /data1/dpopescu/CSV/CMS/CMS-100ev-qcd-s2-s.csv – function granularity
  - Duration: 136.56s

Ttbar example - 100 events

- Step 1 (generation, simulation, digitization) –
  Tb5 file name: /data1/dpopescu/CMS/CMS100ev-s1-ttbar/ttbar100ev-s1.tb5 -- corrupted
- Step 2 (reconstruction) –
  Tb5 file name: /data1/dpopescu/CMS/CMS100ev-s2-ttbar/cmsttbar100ev-s2.tb5

The bash scripts used for monitoring CMS are `cmsrunminbias.sh`, `cmsrunqcd.sh` and `cmsrunttbar.sh`. `Usage: cmsrunminbias.sh -n number_of_events -m factor -o output_path`. The scripts provide log files, automatic directory organization and pinning (taskset). The –m option refers to the possibility of multiplying the sampling periods for the events with a factor of 10 or 100. The scripts can be found in the directory `/disk1/cmssw/work/CMSSW_3_7_0_pre5/src`. .Before running the scripts, the environment has to be set up as described in the Appendix's section dedicated to CMS.

5.6.4 LHCb

First example:

1. 1 event :
   - Gauss –
     - Tb5 file name: /data1/dpopescu/LHCb/Gauss1ev/gauss1ev.tb5
     - CSV file name: /data1/dpopescu/LHCb/CSV/Gauss/Gauss-1ev-ex1-f.csv – function granularity
       /data1/dpopescuLHCb/CSV/Gauss/Gauss-1ev-ex1-s.csv – source granularity
     - Duration: 144.58s

- Boole –
  - Tb5 file name: /data1/dpopescu/LHCb/Boole1ev/boole-1-digi.tb5
  - CSV file name: /data1/dpopescu/CSV/Boole/Boole-1ev-f.csv– function granularity
    /data1/dpopescu/CSV/Boole/Boole-1ev-s.csv– source granularity
  - Duration: 44.06s
- Brunel -
  - Tb5 file name: /data1/dpopescu/LHCb/Brunel1ev/brunel-1-reco.tb5
  - CSV file name: /data1/dpopescu/CSV/Brunel/Brunel-1ev-f.csv– function granularity
    /data1/dpopescu/CSV/Brunel/Brunel-1ev-s.csv – source granularity
  - Duration: 43.2s

2. 50 events :
   - Gauss –
     - Tb5 file name: /data1/dpopescu/LHCb/Gauss50ev/gauss50ev.tb5
     - CSV file name: /data1/dpopescu/CSV/Gauss/Gauss-50ev-ex1-f.csv – function granularity
       /data1/dpopescu/CSV/Gauss/Gauss-50ev-ex1-s.csv – source granularity
     - Duration: 902.18s
   - Boole –
     - Tb5 file name: /data1/dpopescu/LHCb/Boole50ev/boole-50-digi.tb5
     - CSV file name: /data1/dpopescu/CSV/Boole/Boole-50ev-f.csv– function granularity
       /data1/dpopescu/CSV/Boole/Boole-50ev-s.csv– source granularity
     - Duration: 51.98s
   - Brunel -
     - Tb5 file name: /data1/dpopescu/LHCb/Brunel50ev/brunel-50-reco.tb5
     - CSV file name: /data1/dpopescu/CSV/Brunel/Brunel-50ev-f.csv– function granularity
       /data1/dpopescu/CSV/Brunel/Brunel-50ev-s.csv – source granularity
     - Duration: 57.8s

3. 150 events :
   - Gauss –
     - Tb5 file name: /data1/dpopescu/LHCb/Gauss150ev/gauss150ev.tb5
     - CSV file name: /data1/dpopescu/CSV/Gauss/Gauss-150ev-ex1-f.csv – function granularity
       /data1/dpopescu/CSV/Gauss/Gauss-150ev-ex1-s.csv – source granularity
     - Duration: 2408.96s
   - Boole –
     - Tb5 file name: /data1/dpopescu/LHCb/Boole150ev/boole-150-digi.tb5
     - CSV file name: /data1/dpopescu/CSV/Boole/Boole-150ev-f.csv– function granularity
       /data1/dpopescu/CSV/Boole/Boole-150ev-s.csv– source granularity
     - Duration: 68.72s
   - Brunel -
     - Tb5 file name: /data1/dpopescu/LHCb/Brunel150ev/brunel-150-reco.tb5
     - CSV file name: /data1/dpopescu/CSV/Brunel/Brunel-150ev-f.csv– function granularity
       /data1/dpopescu/CSV/Brunel/Brunel-150ev-s.csv – source granularity
     - Duration: 84.84s
4. 500 events:

- Gauss – Tb5 file name: /data1/dpopescu/LHCb/Gauss500ev/gauss-500-sim.tb5 –corrupted tb5 file

Second example:

1. 1 event :
   - Gauss –
     - Tb5 file name: /data1/dpopescu/LHCb/Gauss-ex2-1-sim/gaussex2-1-sim.tb5
     - Duration: 231.02s
   - Boole –
     - Tb5 file name: /data1/dpopescu/LHCb/Boole1evex2/boole-1-digi.tb5
     - CSV file name: /data1/dpopescu/CSV/Boole/boole-1ev-ex2-f.csv.– function granularity
       /data1/dpopescu/CSV/Boole/boole-1ev-ex2-s.csv– source granularity
     - Duration: 74.1s

2. 50 events :
   - Gauss –
     - Tb5 file name: /data1/dpopescu/LHCb/Gauss-ex2-50/gauss-ex2-50-sim.tb5
     - CSV file name: /data1/dpopescu/CSV/Gauss/Gauss-50ev-ex1-f.csv – function granularity
       /data1/dpopescu/CSV/Gauss/Gauss-50ev-ex1-s.csv – source granularity
     - Duration: 2393.56s
   - Brunel -
     - Tb5 file name: /data1/dpopescu/LHCb/BRUNEL-50-RECO2/brunel-50-ex2-reco.tb5
     - Duration: 107.41s

3. 250 events :
   - Gauss –
     Tb5 file name: /data1/dpopescu/LHCb/Gauss-ex2-250/gauss-ex2-250-sim.tb5 --corrupted
   - Boole –
     - Tb5 file name: /data1/dpopescu/LHCb/Boole250ex2/boole-250-digi.tb5
     - Duration: 68.72s
   - Brunel -
     - Tb5 file name: /data1/dpopescu/LHCb/BRUNEL-250-RECO2/brunel-250-ex2-reco.tb5
     - Duration: 83.08s

The Python scripts editGauss.py, editBoole.py, editBrunel1.py (for the first example) and editBrunel2.py (for the second example) are used for changing the number of events and the name of the resulted files. In these scripts, the EventSelector.Input() is set to the directory which contains the necessary files for running the application, so if this changes, the scripts should be updated. The Python scripts are called from the bash scripts runGauss1.sh (for the first example), runGauss2.sh (for the second example), runBoole1.sh (for the first example), runBoole2.sh(for the second example), runBrunel1.sh(for the first example) and runBrunel2.sh(for the second example), which can be used with the following options: `Usage: runGauss1.sh -n number_of_events -m factor -o output_path –i file_name.` They provide log files, automatic directory organization and pinning (taskset). The –m option refers to the possibility of multiplying the sampling periods for the events with a factor of 10 or 100. The –i option is used for specifying the name of the output file for the simulation. This file will be used in the second phase, digitization, and the file resulted from digitization, will be used in reconstruction. The scripts can be found in the directory

`/afs/cern.ch/user/d/dpopescu/cmtuser/`.Before running the scripts, the environment for each application has to be set up as described in the Appendix's section dedicated to LHCb.