News on iLCSoft and iLCDirac

CLIC Week 2019

Marko Petrič

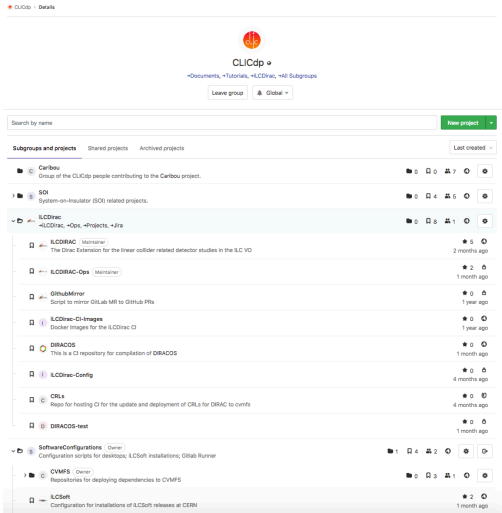
On behalf of the CLICdp collaboration

Geneva, 22 January 2019

Table of Contents

GitLab

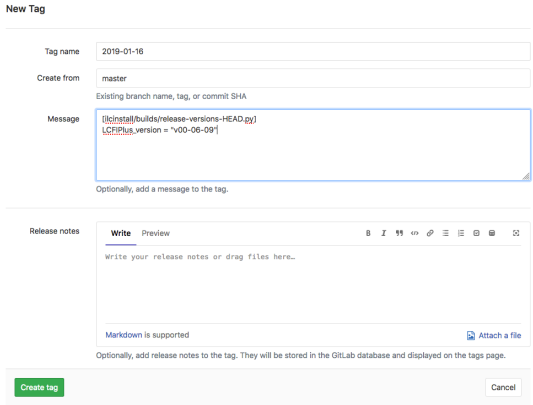
- GitLab group for CLICdp https://gitlab.cern.ch/clicdp
- Usable for all types of software projects, websites, and documents



Continuous Integration

Using GitLab-CI beyond testing pull requests (but also for that of course!)

- “Single Click” to compile and deploy software release on CVMFS, automatic nightly releases
- Deploy websites with auto-generated content: iLCDIRAC documentation, DD4hep
- Compile software for use on the grid using a fixed environment

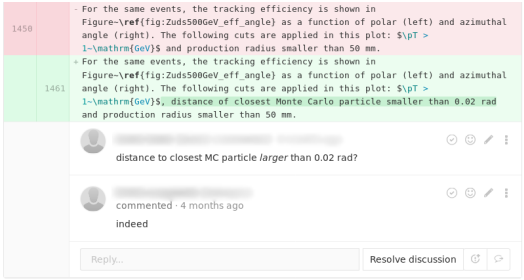


Collaborative Editing

GitLab offers all the tools for collaborative editing, even with just a web browser

- Versioning, access control, automatic builds, web IDE
- https://gitlab.cern.ch/CLICdp/Publications
- You can find all the figures for the publications here

Overleaf is also available for collaborative editing, but not clear how much the versioning helps when things go wrong.



Discuss changes before merging. Only read the phrases that are actually changed

AFS

- AFS is being phased out at CERN at the end of LS2
- `/afs/cern.ch/eng/clic/*` has mostly been archived to `Castor`
- Some harder to relocate software has so-far remained
- Only relevant script:
  - sudo /afs/cern.ch/eng/clic/software/scripts/installCVMFS.sh

CVMFS

All new software is provided via CVMFS: `/cvmfs/clicdp.cern.ch`

- compilers (gcc 6, 7, 8, llvm), iLCSoft, iLCDirac, git, Emacs, Whizard2
- Mounted on desktops, lxplus, lxbatch, and grid sites around the world
- For SL6 and CentOS7, and a few things for macOS

EOS

- Shared storage on EOS `/eos/experiment/clicdp/*`: `grid`, `data`, `phys`
  - `grid`: equal to `CERN-DST-EOS` iLCDirac Storage Element, allow direct **read** access for grid files: both central production and user output
  - `data`: test beam data
  - `phys`: Lumi spectra, background files, MC samples
- Personal storage (1TB) under `/eos/user/u/username/`, also connected to cernbox.cern.ch
- Install access on local machine:
  - sudo /afs/cern.ch/eng/clic/software/scripts/installEOS.sh

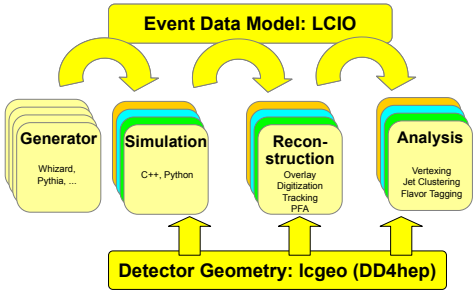
Generation with Whizard2

- Whizard 2.6.3 available in CVMFS
  - source /cvmfs/clicdp.cern.ch/software/WHIZARD /2.6.3/x86_64-slc6-gcc7-opt/setup.sh
- Provide current and future versions relevant to CLIC analyses
- Provide also CIRCE files
  - /cvmfs/clicdp.cern.ch/software/WHIZARD/circe_files/CLIC/
- Run: whizard your_file.sin

```
!Model and Process block
model = SM
process decay_proc = "A", "A" => "b", "B"
?vis_channels=true

compile

!Beam block
sqrts = 350 GeV
!use Circe2_file for the beam spectrum
beams = A, A => circe2
$circe2_file = "0.35TeVggMapPB0.67E0.0Mi0.0.circe"
$circe2_design = "CLIC/GG"
?circe2_polarized = false
?keep_beams=true
?isr_recoil = false
!isr_order = 1

!cuts block
cuts = all E > 10 GeV and Theta > 10 degree and Theta < 170 degree ["b":"B"]

?ps_fsr_active = true
?ps_isr_active = false
?hadronization_active = true
$shower_method = "PYTHIA6"
!?ps_PYTHIA_verbose = true

integrate (decay_proc)

n_events = 10
seed = 1299872493

simulate (decay_proc) {
        $sample = "decay_proc_101-TFF-pol0_cuts_350.001"
        sample_format = stdhep
        $extension_stdhep = "stdhep"
}
```

Linear Collider Software 1/2

- Linear collider community has used and developed **common software** for many years
  - Event data model (EDM) and persistency: LCIO
    - podIO is being investigated in AIDA2020
- Adopted DD4hep geometry description to develop more common software geometry information
- Available@http://github.com/iLCSoft/



Linear Collider Software 2/2

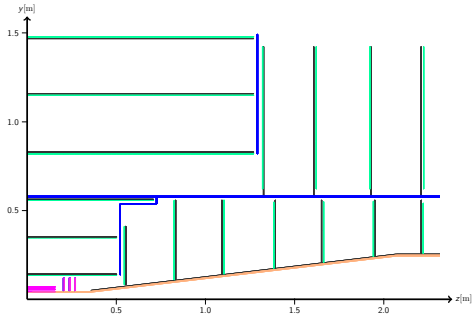
- Regular builds of software available
  /cvmfs/clicdp.cern.ch/iLCSoft/builds/

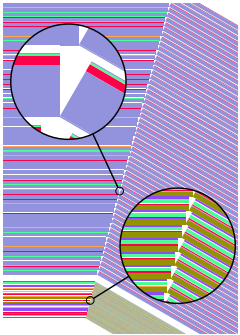
```
[2016-02-02  2016-09-27  2017-02-17  2017-06-21  2017-10-20  2018-02-08  2018-05-18  2018-11-01
 2016-04-06  2016-11-09  2017-03-28  2017-07-04  2017-11-06  2018-03-02  2018-06-29  2019-01-16
 2016-06-22  2016-11-22  2017-04-25  2017-07-12  2017-11-09  2018-04-10  2018-07-24  current
 2016-07-04  2016-11-24  2017-05-15  2017-07-27  2017-11-15  2018-04-26  2018-08-10  nightly
 2016-08-22  2016-12-15  2017-05-30  2017-08-23  2017-12-13  2018-05-14  2018-10-11
 2016-09-12  2017-02-02  2017-06-13  2017-10-14  2017-12-21  2018-05-17  2018-10-26
```

  - Latest build 2019-01-16
  - Always also a most recent build available called nightly
- To initialize a version source the init file e.g.
  source /cvmfs/clicdp.cern.ch/iLCSoft/builds/2019-01-16/x86_64-slc6-gcc62-opt/init_ilcsoft.sh
- Let us know if you have any specific requirements

Detector Model

- Geometry of final detector model implemented: CLIC_o3_v014
- Geometry in detail in CLICdp-Note-2017-001
- Performance in detail in CLICdp-Note-2018-005





Simulation using DD4hep

- `ddsim` python executable is part of the DD4hep release
- Get steering file `ddsim --dumpSteeringFile > mySteer.py`
  - Steering file includes documentation for parameters and examples
  - The python file contains a `DD4hepSimulation` object at global scope
  - Configure simulation directly from commandline

```
from DDSim.DD4hepSimulation import DD4hepSimulation
from SystemOfUnits import mm, GeV, MeV, keV
SIM = DD4hepSimulation()
SIM.compactFile = "CLIC_o3_v06.xml"
SIM.runType = "batch"
SIM.numberOfEvents = 2
SIM.inputFile = "electrons.HEPEvt"
SIM.part.minimalKineticEnergy = 1*MeV
SIM.filter.filters ['edep3kev'] =
dict (name="EnergyDepositMinimumCut/3keV" ,
      parameter={"Cut" : 3.0*keV} )
SIM.action.tracker = "Geant4TrackerWeightedAction"
SIM.action.calo = "Geant4ScintillatorCalorimeterAction"
```

```
$ ddsim
--action.calo
--action.mapActions
--action.tracker
--compactFile
--crossingAngleBoost
--dump
--dumpParameter
--dumpSteeringFile
--enableDetailedShowerMode
--enableGun
--field.delta_chord
--field.delta_intersection
--field.delta_one_step
--field.eps_max
--field.eps_min
--field.equation
--field.largest_step
--field.min_chord_step
--field.stepper
--filter.calo
--filter.filters
--filter.mapDetFilter
--filter.tracker
-G
--gun.direction
--gun.energy
--gun.isotrop
--gun.multiplicity
--gun.particle
--gun.position
-h
--help
-I
--inputFiles
-M
--macroFile
-N
--numberOfEvents
-O
--outputFile
--output.inputStage
--output.kernel
--output.part
--output.random
--part.keepAllParticles
--part.minimalKineticEnergy
--part.printEndTracking
--part.printStartTracking
--part.saveProcesses
--physics.decays
--physics.list
--physicsList
--physics.rangecut
--printLevel
--random.file
--random.luxury
--random.replace_gRandom
--random.seed
--random.type
--runType
-S
--skipNEvents
--steeringFile
-v
--vertexOffset
--vertexSigma
```

Simulation with standard parameters

- Standard configurations:
  https://github.com/iLCSoft/CLICPerformance
- Install location:
  /cvmfs/clicdp.cern.ch/iLCSoft/builds/<date>/ClicPerformance/HEAD
- Simulation events from an input file
- Example:

```
ddsim --compactFile=__path__/CLIC_o3_v14.xml
  --inputFile __path__/input.stdhep -N 2
  --outputFile=my_output.slcio
  --steeringFile=__path__/clicConfig/clic_steer.py
```

Reconstruction with standard parameters

- All configuration also in the same package – CLICPerformance
- Example:

```
Marlin --InitDD4hep.DD4hepXMLFile=__path__/CLIC_o3_v14.xml
  --global.LCIOInputFiles=my_sim.slcio clicReconstruction.xml
  --Config.Tracking=Conformal
  --Output_REC.LCIOOutputFile=my_rec.slcio
  --Output_DST.LCIOOutputFile=my_dst.slcioy
```

- Example with overlay:

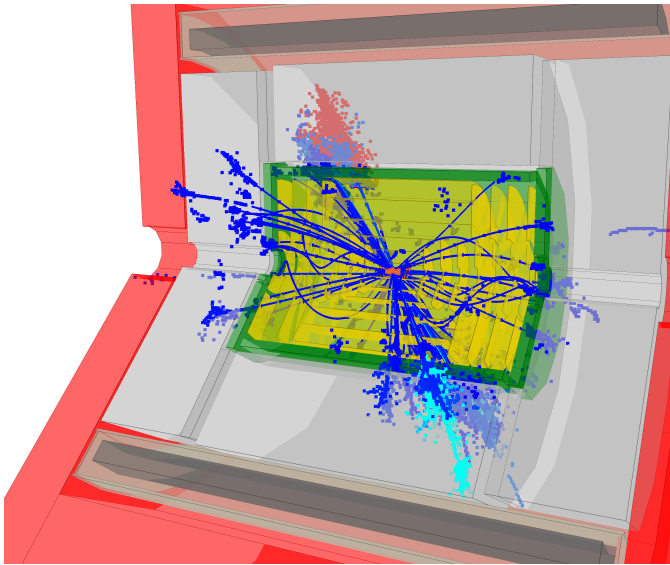
```
Marlin --InitDD4hep.DD4hepXMLFile=__path__/CLIC_o3_v14.xml
  --global.LCIOInputFiles=my_sim.slcio clicReconstruction.xml
  --Config.Tracking=Conformal
  --Config.Overlay=3TeV
  --Overlay3TeV.BackgroundFileNames=BACKGROUND_FILE.slcio
  --Output_REC.LCIOOutputFile=my_rec.slcio
  --Output_DST.LCIOOutputFile=my_dst.slcio
```

Reconstruction run-time Performance

- Simulation and Reconstruction chain finalized
  - Improvements in reconstruction time
  - Improvements in memory management
- Automated check of memory leaks via valgrind
- Tracking run-time/hot-spot checks with “Intel VTune Amplifier”
- Reconstruction time for 3 TeV ttbar event: ~15-20 min/event
- Reconstruction takes ~ 5 times longer than simulation

Event Display

- Using CEDVierer part of iLCSoft
- Special visualization geometry CLICPerformance/Visualisation/CLIC_o3_v06_CED/
- Example: `ced2go -d CLIC_o3_v06_CED.xml -t ced2go-template-DD4.xml -s 1 my_rec.slcio`



Event Display settings

- configuration of visualization in ced2go-template-DD4.xml
  - Color properties
  - Visualized collections

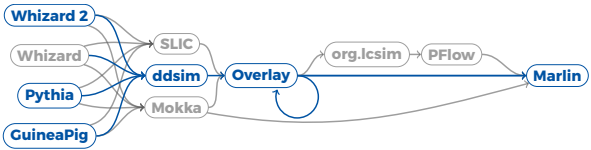
```
<!-- Color recunstructed particle by energy -->
<parameter name="ColorByEnergy" type="bool">true</parameter>
<!-- Minimal value for energy which will be represented as blue -->
<parameter name="ColorByEnergyMin" type="double">0.0</parameter>
<!-- Maximal value for energy which will be represented as red -->
<parameter name="ColorByEnergyMax" type="double">35.0</parameter>
<!-- Hue value that will be used to determine the pallete -->
<parameter name="ColorByEnergySaturation" type="double">1.0</parameter>
<!-- Brigtness value that will be used to determine the pallete -->
<parameter name="ColorByEnergyBrightness" type="double">1.0</parameter>
<!-- Automatically adjust event by event the blue to min energy and red to max energy of event -->
<parameter name="ColorByEnergyAutoColor" type="bool">true</parameter>
<!-- Scale line thickness of drawn helixes, usefull for image dumps -->
<parameter name="ScaleLineThickness" type="double">12</parameter>
<!-- Scale marker size of cluster markers, usefull for image dumps -->
<parameter name="ScaleMarkerSize" type="double">1</parameter>
```

```
<parameter name="DrawInLayer">
      PandoraPFOs 0 3 3
      LE_LooseSelectedPandoraPFOs 0 3 4
      LE_SelectedPandoraPFOs 0 3 5
      LE_TightSelectedPandoraPFOs 0 3 6
      LooseSelectedPandoraPFOs 0 3 7
      SelectedPandoraPFOs 0 3 8
      TightSelectedPandoraPFOs 0 3 9
```

iLCDirac Use Case

- ILC VO: virtual organization for linear colliders (CLIC and ILC)
- iLCDirac is an extension of the DIRAC system for the ILC VO
- Centralized MC Production and User jobs
- Capacity: around 15k to 20k job slots available at best of times

- Define application payload via interfaces
- Chain applications (append one after the other)



To initialize: source /cvmfs/clicdp.cern.ch/DIRAC/bashrc

Whizard2 interface 1/2

- Whizard 1.9.5 not ideal for on-the-fly configuration
  - Need to compile configuration
- Whizard 2.0.0. and onwards allows to steer generation

```
!Model and Process block
model = SM
process decay_proc = "A", "A" => "b", "B"
?vis_channels=true

compile

!Beam block
sqrts = 350 GeV
!use Circe2_file for the beam spectrum
beams = A, A => circe2
$circe2_file = "0.35TeVggMapPB0.67E0.0Mi0.0.circe"
$circe2_design = "CLIC/GG"
?circe2_polarized = false
?keep_beams=true
?isr_recoil = false
!isr_order = 1

!cuts block
cuts = all E > 10 GeV and Theta > 10 degree and Theta < 170 degree ["b":"B"]

?ps_fsr_active = true
?ps_isr_active = false
?hadronization_active = true
$shower_method = "PYTHIA6"
!?ps_PYTHIA_verbose = true

integrate (decay_proc)

n_events = 10
seed = 1299872493

simulate (decay_proc) {
        $sample = "decay_proc_101-TFF-pol0_cuts_350.001"
        sample_format = stdhep
        $extension_stdhep = "stdhep"
}
```

Whizard2 interface 2/2

- iLCDirac interface to use Whizard on the grid
  - Installations provided centrally via CVMFS including CIRCE files
- Caveats for your sin file:
  - Set seed, number of events and output file via iLCDirac API!
  - The decay process has to be named `decay_proc`
  - The simulate block is automatically added by iLCDirac

```
from ILCDIRAC....Applications import Whizard2

whiz = Whizard2()
whiz.setSinFile("sample.sin")
whiz.setSeed(1234)
whiz.setNumberOfEvents(30)
whiz.setOutputFile("generated.stdhep")
```

```
process decay_proc = "A", "A" => "b", "B"
#Code inserted by setSinFile
#Appended by iLCDirac
n_events = "iLCDirac API"
seed = "iLCDirac API"

simulate (decay_proc) {
        $sample = "iLCDirac API"
        sample_format = iLCDirac API
        $extension_stdhep = "iLCDirac API"
}
```

Sim and Reco jobs

- Extensive documentation available at http://lcd-data.web.cern.ch/lcd-data/doc/ilcdiracdoc/
- Including examples for each step
  - Running Marlin
  - Running DDSim
  - Running DDSim and then Marlin
  - Running Overlay and Marlin
  - Running Overlay and Marlin with CLIC_o3_v12
  - Automatic Job Splitting
  - Follow this link

Job splitting

- Create multiple jobs with similar configuration
- Easily split the workload over several jobs
  - use `setSplitEvents` and specify number of jobs and events/job

```
job = UserJob()
job.setOutputSandbox( "*.log" )
## output auto-changed to, e.g., ddsimout_5.slcio
job.setOutputData( "ddsimout.slcio", outputPath="sim1" )
job.setCLICConfig( "ILCSoft-2017-07-27" )
## creates 10 jobs with 100 events each
job.setSplitEvents( eventsPerJob=100, numberOfJobs=10 )

ddsim = DDSim()
ddsim.setVersion("ILCSoft-2017-07-27_gcc62")
ddsim.setDetectorModel("CLIC_o3_v13")
ddsim.setExtraCLIArguments( " --enableGun --gun.particle=mu- " )
ddsim.setNumberOfEvents( 100 )
ddsim.setSteeringFile( "clic_steer.py" )
ddsim.setOutputFile( "ddsimout.slcio" )
myJob.append(ddsim)
myJob.submit( dllc )
```

Support

- In case of fire:
  1. Consult documentation:
     http://lcd-data.web.cern.ch/lcd-data/doc/ilcdiracdoc/
  2. Before submitting a ticket, see: http://lcd-data.web.cern.ch/lcd-data/doc/ilcdiracdoc/DOC/Files/UserGuide/support.html
     - Submit a ticket to the issue tracker
       https://its.cern.ch/jira/browse/ILCDIRAC
     - Or send an email: ilcdirac-support@cern.ch

Please remember this, and also remind your supervisees and colleagues