CURRENT STATUS AND OUTLOOK

CLIC Detector and Physics Collaboration Meeting

Marko Petrič

Genève, 30 August 2017

Contents

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
  - Workflow Modules for LC Software, Overlay System
  - JPCS. ILCDirac, a DIRAC extension for the Linear Collider community. Proceedings of CHEP2013. 513 CLICdp-Conf-2013-003
  - JPCS. Using OSG Computing Resources with (iLC)DIRAC. Proceedings of CHEP2016. CLICdp-Conf-2017-003
- Centralized MC Production (Event Generation, Sim and Rec)
- User jobs (Generation, Simulation, Reconstruction, Analyses)

Capacity:

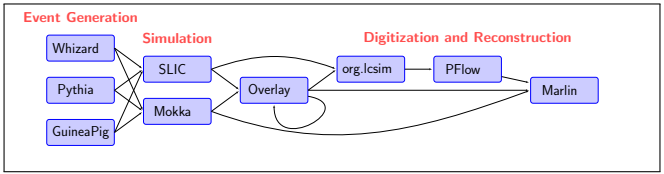
- Using WLCG and OSG resources (Globus, Arc, HTCondorCE)
  - Mostly opportunistic, some dedicated
  - Around 15k to 20k job slots available at best of times

Code: https://gitlab.cern.ch/CLICdp/iLCDirac/ILCDIRAC

API, Workflow

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

```
from DIRAC.Core.Base import Script
Script.parseCommandLine()
import UserJob
import Marlin
import DiracILC
d = DiracILC()
j = UserJob()
j.setOutputData("recEvents.slcio")
m = Marlin()
m.setVersion("ILCSoft-01-17-09")
m.setSteeringFile("Steering.xml")
m.setInputFile("SimEvents.slcio")
j.append(m)
j.submit(d)
```



Server Setup

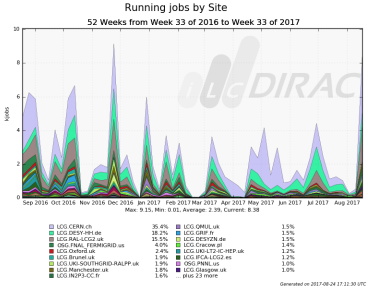
Using set of redundant servers for backup, in case primary servers go down

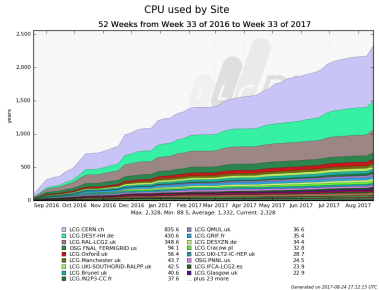
Total of 100 Cores and 200 GB of Ram, SLC6 Virtual Machines

- 2 × 3 Servers running Agents and Services: 8 Cores, 16 GB RAM; Split by DIRAC-System
  1. Framework, Transformation, DataManagement, Configuration
  2. StorageManagement, WorkloadManagement
  3. RequestManagement, Accounting, ResourceStatus
- 3 DIRAC DIP-Storage SEs: 4 Cores, 8 GB Ram, 1 TB Volume
  - DIP-SE, Log-SE, SB-SE
- Web Server 4 Cores, 8 GB RAM
- DBs hosted on CERN DB on Demand (iLCDirac, ilcacdb (accounting DB), ilcdtest)
- Development, Testing, continuous integration (8 × 1 core), and spare VMs

CPU Usage

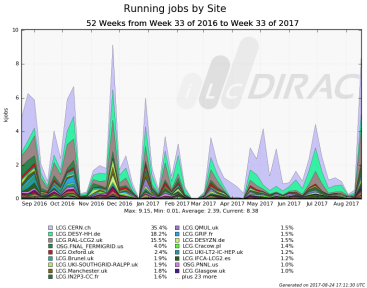
- Activity in bursts
- Maximum > 10k jobs
- Integrated all OSG resources allowing ILC-VO
  - HTCondor-CE and Globus Computing Elements
  - No SiteDirectors at Sites
- Slightly less resources used in the last year compared to previous

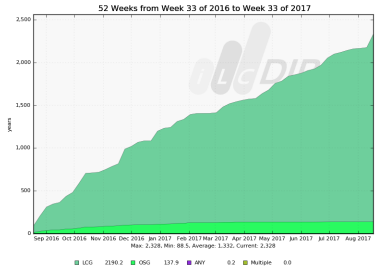




CPU Usage

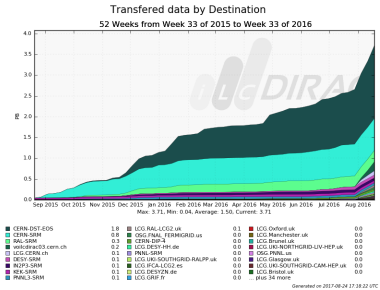
- Activity in bursts
- Maximum > 10k jobs
- Integrated all OSG resources allowing ILC-VO
  - HTCondor-CE and Globus Computing Elements
  - No SiteDirectors at Sites
- Slightly less resources used in the last year compared to previous

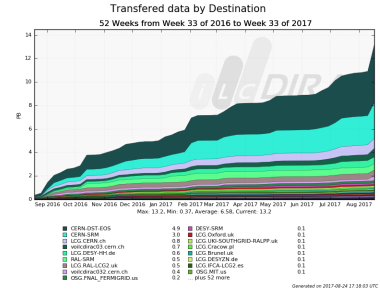




DataManagement

- Using the DiracFileCatalog
- 27 Million files (32 million replicas), 5.9 PB (6.3 PB total), 10 Million files, 2.2 PB since the last workshop
- Metadata used to define input files for transformations
- Heavier usage of EOS → more transferred data
- All based on XRoot





Testing iLCDirac

- Running Continuous Integration for iLCDirac via GitlabCI+Travis
- Using 4 categories of tests on SLC6 and CC7
  - Workload tests: try execution of job tests cases
  - SE tests: try copy/add/remove between SEs (XRoot,SRM)
  - unit tests: code fragments to test individual functions
  - pylint: require full pylint conformity (no errors!)
- Using the HEAD of DIRAC release branch (rel-v6r15)
- Installation of iLCDirac on SL6 and CC7
- Aiming for as complete coverage as possible in iLCDirac
  - Catch bugs in our code
  - Catch interface changes in DIRAC
- Gitlab creates PR on Github to run landscape (health 96%) and unit tests on Travis

Testing iLCDirac evolution

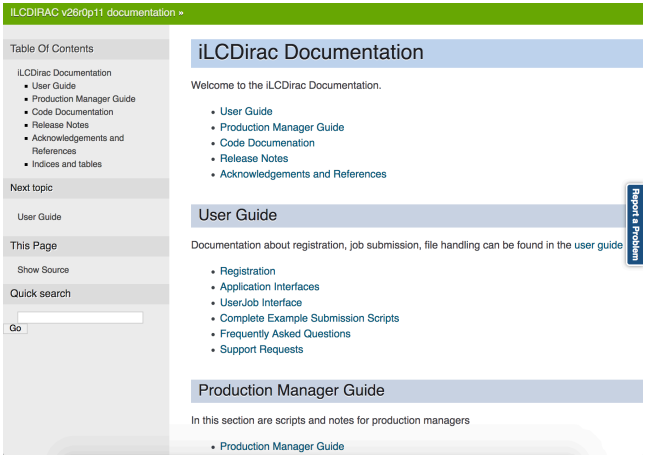
- Heavily invested in extending tests since last workshop

test type	May 2016	August 2017
Workload	9	9 (SL6&CC7)
SE	0	3 (SL6&CC7)
unit	280	1322
pylint	✓	✓

- Increased unit test coverage from 36% to 58%
- From 6615/17007 relevant lines to 9435/16390 relevant lines covered
- Credit to Jan Ebbing

iLCDirac Documentation

- Using sphinx based documentation
- Linked with DIRAC code documentation for base classes, functions,...
- Documenting the iLCDirac API for application configuration
- http://lcd-data.web.cern.ch/lcd-data/doc/ilcdiracdoc/
- Plan to consolidate all documentation in sphinx and deprecate TWiki



Operational developments

- Some developments for iLCDirac
  - Improvement of the DataRecoveryAgent
  - Adoption of VOMS2CS
  - Transformations to move only files with descendants
    - new plugin BroadcastProcessed
  - Access to CC7 site (Edinburgh)
- Some contributions to DIRAC
  - Improvements to HTCondor-CE
  - Multiple operations in Data Operation Transformations
  - work on FileCatalog
    - descendants/ancestors
    - slow recursive operations
  - Improvements of Documentation (Agent Parameters, Code Doc)
  - Bugfixes

Job splitting (Technology preview)

To come with update to Dirac v6r17 (19...)

- 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 )
```

New production system (Technology preview)

- Old production syst. based on very long “complicated” `.py` file
  - Hard to see at a glance what the configuration of the production is
- Redesign the production creation based on configuration file
- Easy overview and steering of productions

```
[Production Parameters]
prodGroup = several
detectorModel = CLIC_o3_v12
softwareVersion = ILCSoft-2017-07-27_gcc62
clicConfig = ILCSoft-2017-07-27
eventsPerJobs =

energies =
processes =
## optional prodid to search for input files
# prodIDs =

## number of events for input files to split productions
NumberOfEventsInBaseFiles =

productionLogLevel = VERBOSE
outputSE = CERN-DST-EOS

finalOutputSE = CERN-SRM

## optional additional name
# additionalName = None

#Productions to create: Gen, Split, Sim, Rec, RecOver
ProdTypes =

move = False

#Datatypes to move: Gen, Sim, Rec, Dst
MoveTypes =
```

Whizard2 interface 1/2 (Technology preview)

- 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 (Technology preview)

- 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"
}
```

iLCDirac: Support

- If 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

Summary

- iLCDirac running stable
- New production chain will be deployed as soon as it is tested thoroughly
- Continuing effort for high availability, usability, efficiency