



Linear Collider Software at CLIC

André Sailer

On behalf of the CLICdp collaboration

CERN-EP-LCD

LCWS 2017

Strasbourg

October 24, 2017

Table of Contents



1 Simulation and Reconstruction

2 Infrastructure

3 Beyond the Next Mass Production

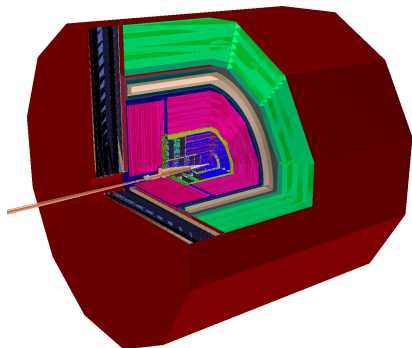
4 Summary

Section 1:



1 Simulation and Reconstruction

- Detector Model frozen, documented in note
- Thirteen is our lucky number



Reconstruction workflow

- 1 Overlay
 - 2 Digitisation
 - 3 Track Pattern recognition (TruthTracking or ConformalTracking)
 - 4 Track Fit (Refit for better track parameter estimate)
 - 5 Particle Flow Reconstruction (PandoraPFA)
 - 6 Forward calorimeter reconstruction (LumiCal/BeamCal)
 - 7 PFO selection
- Implemented in one unified steering file for the reconstruction that can be configured on the fly: avoid duplicating parameter settings in different files that will diverge as much as possible
`CLICPerformance/examples/clicReconstruction.xml`
 - `Marlin --Config.Tracking=Conformal --global.LCIOIn...`

- The reconstruction is running from overlay to flavour tagging, etc.
 - ▶ Centrally creating larger sized samples: $Z \rightarrow uds, qqqq, \tau\bar{\tau}$ in iLCDirac
 - ▶ with and without overlay of $\gamma\gamma \rightarrow \text{hadron}$
- Performances are being studied to identify issues that still need to be addressed, can, or should be improved
 - ▶ Improve pattern recognition for displaced tracks, tune parameter settings
 - ▶ Improve track parameter resolutions, parameter uncertainty
 - ▶ Improve particle ID efficiencies
 - ▶ Tune particle flow reconstruction
 - ▶ Add PFO Selector for lower energy

Details on current status of performance in Emilia's talk on tracking and my talk on calorimetry

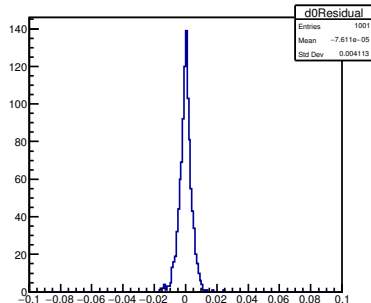
Track Reconstruction



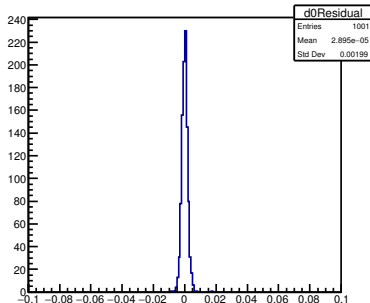
- Pattern Recognition via Conformal Tracking processor for all tracking detectors combined
- Initial TrackFit + Refitting with the output as initial parameters

D0 Resolution before and after refit

First Fit



Refit

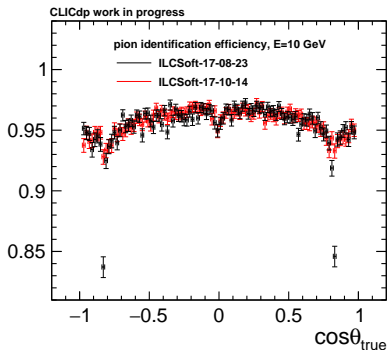


[See Emilia Leogranda's presentation](#) Thursday afternoon for the tracking performance

Finalising Particle Flow reconstruction

- Apply software compensation at higher shower energies
- Improved track cluster matching

[More details](#) on Wednesday morning



Section 2:



2 Infrastructure



<https://github.com/iLCSoft>

- build in features: Pull Request: **code review**, issue tracker
- Extensions: TravisCI, coverity static analysis
 - ▶ Nightly build deployed to /cvmfs/clicdp.cern.ch, used in CIs

ilcsoft.github.io

	Nightly	Build GCC	Build LLVM	
iLCSoft		build passing	build passing	
AIDASoft	Build Status	Issues 6	PRs 1	Coverity Scan
DD4hep	build passing	Issues 0	pull requests 1	coverity passed
aidaIT	build passing	Issues 0	pull requests 0	coverity passed
iLCSoft	Build Status	Issues 1	PRs 1	Coverity Scan
CED	build passing	Issues 1	pull requests 0	coverity passed
CEDViewer	build passing	Issues 1	pull requests 1	coverity passed
CLICPerformance	build passing	Issues 1	pull requests 1	coverity passed

⋮

- Run two-hourly compilation of iLCSoft with `gcc` and `clang`
 - ▶ currently: `gcc 6.2`, `llvm 3.9`
 - ▶ Notification when merge breaks dependent package
- Nightly deployment to `/cvmfs/clicdp.cern.ch` used in CI system on github
 - ▶ CVMFS also includes `git`, `compiler`, `emacs`, `WHIZARD2`, `ILCDIRAC`

Plan to control nightly builds and releases via GitLab-CI system

- Web interface
- Scheduling for nightly builds part of GitLab-CI
- Deploy with single button

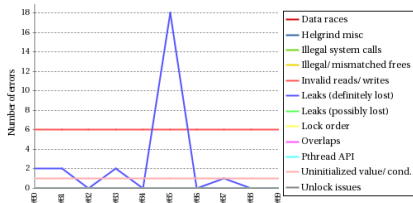
- Individual nightlies to monitor number of compiler warnings
 - ▶ Once package without warnings easy to disallow new ones (-Werror in CI build)
 - ▶ Still large amount of *technical debt*
 - ★ E.g.: MarlinReco has 2200 compiler warnings
 - ★ Some of these might be actual bugs, decrease performance

All		Overview				
S	W	Name	Last Success	Last Failure	Last Duration	# Compiler Warnings
	🔔	aidaTT	37 min - #110	N/A	1 min 19 sec	13
	🔔	CED	17 hr - #110	N/A	23 sec	116
	🔔	CEDViewer	9 hr 25 min - #110	N/A	1 min 3 sec	192
	🔔	ClicPerformance	3 hr 23 min - #137	N/A	31 min	0
	🔔	Clupatra	6 hr 24 min - #109	N/A	1 min 12 sec	77
	🔔	ConformalTracking	13 hr - #108	N/A	52 sec	331
	🔔	DDKalTest	5 hr 3 min - #115	N/A	54 sec	59
	🔔	DDMarlinPandora	11 hr - #106	N/A	1 min 40 sec	0
	🔔	FCalClusterer	8 hr 23 min - #108	N/A	2 min 22 sec	13
	🔔	ForwardTracking	13 hr - #115	N/A	1 min 32 sec	230
	🔔	#CUI	14 hr - #106	N/A	3 min 16 sec	11
	🔔	KalDet	7 hr 15 min - #110	N/A	2 min 14 sec	293
	🔔	KalTest	4 hr 0 min - #107	N/A	1 min 50 sec	115
	🔔	KITrack	5 hr 13 min - #109	N/A	1 min 10 sec	105
	🔔	KITrackMarlin	17 hr - #109	N/A	1 min 24 sec	192
	🔔	LCFIPlus	7 hr 47 min - #111	N/A	2 min 35 sec	6
	🔔	LCFIVertex	14 hr - #110	N/A	4 min 39 sec	4
	🔔	lqgeo	22 hr - #117	N/A	7 min 25 sec	0
	🔔	LCIO	18 hr - #107	N/A	4 min 11 sec	10

Memory Checking



- Nightly run of `valgrind` to monitor memory leaks in reconstruction
- extensive `valgrind` suppression file for DD4hep, ROOT, LCIO false positives
- Fixed all *definite* and *possible* leaks for processors used in CLIC reconstruction chain
- If there is a new leak, receive email notification. Only Look at PRs of last 24 hours



Code Quality and Modernisation



For example the ConformalTracking package

- No more compiler warnings in gcc62 or clang/llvm39
 - Wall, Wextra, Wshadow, Wefc++
- Enforce consistent formatting with clang-format: make format, or git pre-commit hook
- Use unique_ptr, shared_ptr for simplified memory management.
 - unique_ptr forces efficiency by avoiding copies
 - Automatic cleanup

```
copy_if (make_move_iterator (tracksTemp.begin ()),
         make_move_iterator (tracksTemp.end ()),
         back_inserter (tracks),
         [this] (UcellularTrack const& track) {
             return (int (track->size ()) >= (m_minHits - 1)); });
tracksTemp.clear ();
```

Standard algorithm to move objects from one vector to another using unique_ptr and lambda function. No for loop or if/else, delete, etc.

Section 3:



3 Beyond the Next Mass Production

- Modernise software used for linear colliders
 - ▶ Get serious about code quality
 - ★ Enforce uniform formatting, at least be consistent inside each package, file, or pull request
 - ★ no more compiler warnings merged into the repository
 - ▶ C++14 (17?)
 - ▶ Look into multi threaded running of reconstruction
- Too many packages prohibit complete testing for PRs
 - ▶ Would be great to test everything down to running of reconstruction

Let's discuss at the end of this session

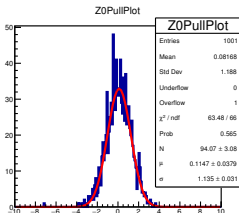
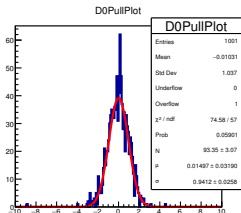
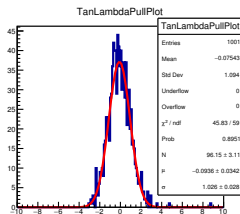
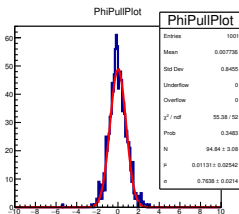
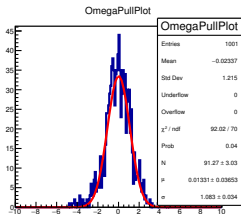
Section 4:



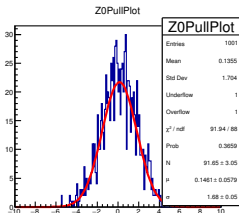
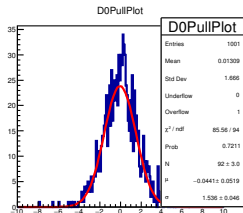
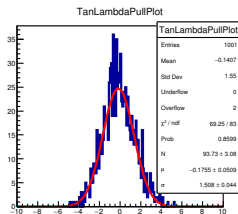
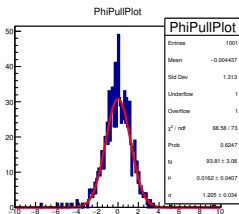
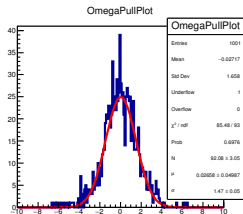
4 Summary

- Pieces for the next large scale production falling into place
 - Finalise detector calibrations and then start mass production
- Automatic testing is absolutely essential, and simplifies one's life, prevents surprises, and let's one do more important and interesting things
- Start thinking about medium term development for software

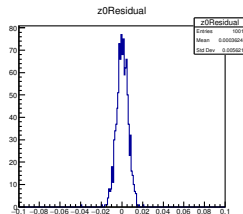
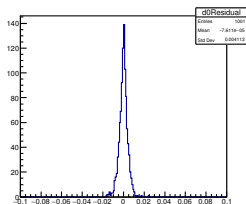
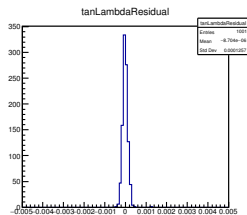
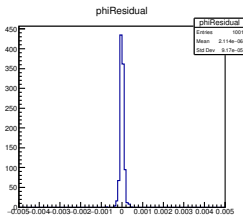
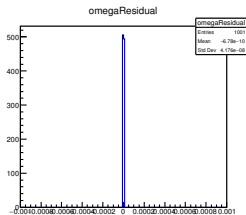
Backup Slides



Pulls Refit



Resolution TruthTrackFinder



Resolution Refit

